September 1989 *Technical Report No. 113*

ONR Final Report

Tools for Simulation-Based Training

Douglas M. Towne
Allen Munro

Behavioral Technology Laboratories
University of Southern California

Sponsored by

Office of Naval Research
Cognitive Science Research Program

Navy Personnel Research and Development Center

Air Force Human Resources Laboratory

Under Contract No. N00014-87-C-0489

Tools for Simulation-Based Training

Technical Report No. 113
ONR Final Report:
Contract No. N00014-87-C-0489

Douglas M. Towne
Allen Munro

Behavioral Technology Laboratories
University of Southern California
1845 South Elena Avenue, Fourth Floor
Redondo Beach, CA 90277

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

1a REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for Public Release: Distribution Unlimited
2b DECLASSIFICATION / DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) Technical Report No. 113	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Behavioral Technology Laboratories - USC 6b OFFICE SYMBOL (If applicable)	7a NAME OF MONITORING ORGANIZATION Cognitive Science Research Programs Office of Naval Research (Code 1142cs)
6c ADDRESS (City, State, and ZIP Code) 1845 S. Elena Ave., 4th Floor Redondo Beach CA 90277	7b ADDRESS (City, State, and ZIP Code) 800 N. Quincy Street Arlington VA 22217-5000
8a NAME OF FUNDING / SPONSORING ORGANIZATIONS Navy Personnel Research and Development Center and Air Force Human Resources Laboratory 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-87-C-0489
8c ADDRESS (City, State, and ZIP Code) Code P-306 San Diego CA 92152	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO: 4428011 (PROJECT NO); TASK NO; WORK UNIT ACCESSION NO
11 TITLE (Include Security Classification) Tools for Simulation Based Training	
12 PERSONAL AUTHOR(S) Douglas M. Towne and Allen Munro	
13a TYPE OF REPORT Final	13b TIME COVERED FROM 6-87 TO 8-89
14 DATE OF REPORT (Year, Month, Day) 89-10-02	15 PAGE COUNT 52
16 SUPPLEMENTARY NOTATION	

17 COSATI CODES		
FIELD	GROUP	SUB-GROUP
05	09	08

18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Artificial Intelligence, Graphical Simulation, Troubleshooting Expertise, Simulation Training, Representing Device Behavior, Diagnostics

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

The Intelligent Maintenance Training System (IMTS) is a set of software tools that permits the composition and presentation of interactive graphical simulations for computer-based technical training. IMTS is designed to support training on the operation and maintenance of complex devices. Simulations are authored by device experts, who use the IMTS tools to draw the components of the device to describe their behavior, and to create simulations made up of the components.

IMTS provides special support for maintenance training. An artificial expert on troubleshooting strategy, called Profile, generates instruction and advice for students. RAPIDS is an additional set of tools, built on the foundation of IMTS, that enables the authoring of a wide variety of simulation-based training courses. Using RAPIDS, an expert creates lessons by performing in the simulation the tasks that are to be taught to students.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL Dr. Susan Chipman	22b TELEPHONE (Include Area Code) (202) 696-4318 22c OFFICE SYMBOL ONR-1142cs

DD Form 1473, JUN 86 Previous editions are obsolete SECURITY CLASSIFICATION OF THIS PAGE

ABSTRACT

The Intelligent Maintenance Training System (IMTS) is a set of software tools that permits the composition and presentation of interactive graphical simulations for computer-based technical training. IMTS is designed to support training on the operation and maintenance of complex devices. Simulations are authored by device experts, who use the IMTS tools to draw the components of the device to describe their behavior, and to create simulations made up of the components.

IMTS provides special support for maintenance training. An artificial expert on troubleshooting strategy, called *Profile*, generates instruction and advice for students.

RAPIDS is an additional set of tools built on the foundation of IMTS, that enables the authoring of a wide variety of simulation-based training courses. Using *RAPIDS*, an expert creates lessons by performing in the simulation the tasks that are to be taught to students.

ACKNOWLEDGEMENTS

The work described here was supported by the Office of Naval Research, the Navy Personnel Research and Development Center, and the Air Force Human Resources Laboratory, under ONR Contract No. N00014-87-C-0489. Susan Chipman served as ONR scientific officer, Vern Malec served as NPRDC scientific officer, and J. Wesley Regian served as AFHRL scientific officer for this contract.

Our colleagues Lee D. Coller, Quentin A. Pizzini, David S. Surmon, and James Wogulis assisted in the design and carried out the implementations of IMTS and RAPIDS.

The term yoking was suggested by Jeffrey Richardson. Subject-matter expertise for the Bladefold simulation was provided by William Johnson. Subject-matter expertise for the WSC-3 Satellite Communication System was provided by Ronald Renfro.

TABLE OF CONTENTS

Tools for Simulation-Based Training

Final Report: ONR Contract N00014-87-C0489

Introduction

The exploitation of interactive graphical simulation for computer-based instruction has been limited by the time and expense typically associated with the production of complex simulations. The Intelligent Maintenance Training System (IMTS) provides an environment for the composition and presentation of such simulations. The authoring environment permits the construction of simulations based on either of two quite different approaches. In one, which is component-oriented, model-based simulations are composed by direct manipulation. In the other, simulations based on the behavior of an equipment system as a whole are built up by creating tables of data that describe that behavior. The former approach is called *deep simulation*; the latter, *surface simulation.*

The model-based, generative approach (deep simulation) has two advantages over a table-look-up style of simulation. First, it permits more robust simulations, which provide nearly complete free-play features. Second, object-oriented models can be developed relatively rapidly since the developer does not have to describe behaviors of the total system. To employ the model-based approach, an author must understand the functions of the objects in the simulated system. An author who understands a complex system only in terms of the behaviors of the system in various operating modes would have difficulty following the component-oriented model-based approach.

Overview of IMTS

IMTS provides editors for composing both deep and surface interactive graphical simulations for training without using computer programming. It also includes a generic expert that can generate instruction in the domain of fault diagnosis. The resulting simulations are presented in an environment that permits students to directly manipulate graphical controls and to observe the effects of these manipulations on simulated indicators and test points.

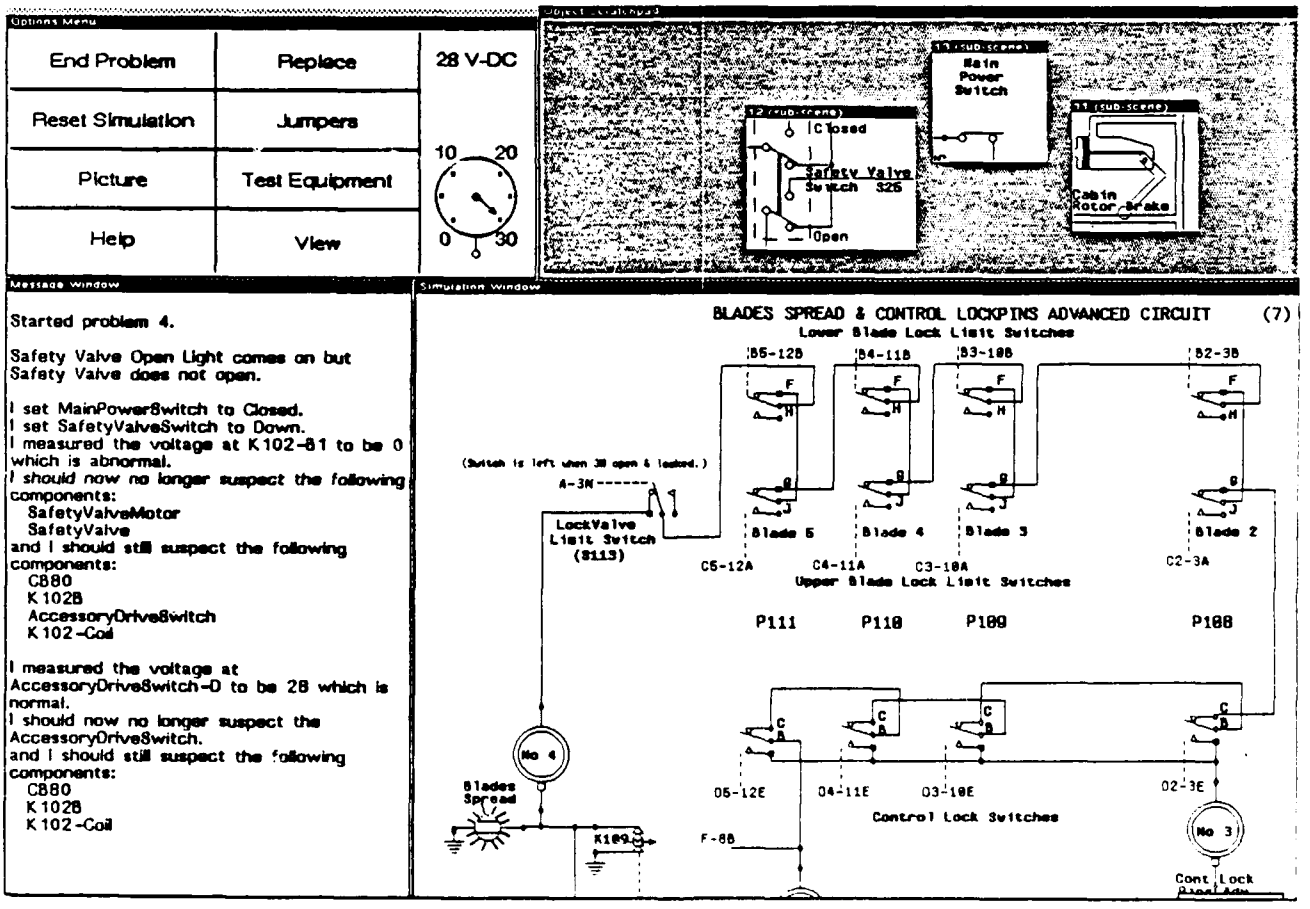
IMTS attacks two productivity problems for the authoring of simulation-based training: (1) the development of flexible and accurate simulations at reasonable cost, and (2) the authoring of expertise about the model domain.

The Student Interface

IMTS simulations may depict the simulated device or system in a variety of different ways, including schematically and in a front-panel format. Large simulations are divided into multiple graphic *simulation scenes*, each of which depicts some portion of the whole device. Students can navigate through the scenes 1) by bringing up a hierarchical map of all the scenes and selecting the one they wish to view, or 2) by selecting special *scene icons* that act as doorways to other scenes.

The student manipulates some of simulated objects by use of the mouse. When the object, such as a switch, changes state in response to the student, it correspondingly changes its appearance, and it usually causes other objects to change their states. In addition to manipulating controls and observing simulated front panel indicators and internal actions of objects, students can examine values at object ports using simulated test equipment. When they work with large simulations, students sometimes discover behaviors of which even the authors were not aware.

The figure below shows the entire IMTS screen during student use.



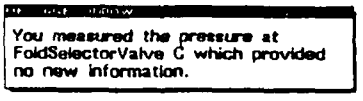
The largest window displays any one scene in the simulation. The scene shown depicts a portion of a helicopter blade-folding system. At this time the IMTS is explaining, in the left-hand text window, how an expert would have approached a just-completed problem. Also, the student had made copies of three objects from other scenes, and placed them in the upper scratch-pad area. These duplicate copies are manipulable and graphically dynamic, just as are the originals from the other scenes.

Generation of Domain Expertise

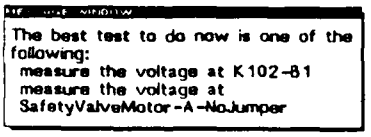
In contrast with the use of conventional expert systems methodology, IMTS does not require the authoring of expertise about troubleshooting a particular device. Instead, a generic troubleshooting expert, called Profile (Towne, 1984, 1986; Towne & Johnson, 1987), is applied to data generated from the simulation model to produce evaluations of student actions, recommendations and other advice, and normative or expert solutions. Profile's use of simulation-generated data is an example of an approach we have sought to apply wherever possible in IMTS: to exploit the model data and the simulation as fully as possible to **generate** instruction rather than requiring expensive authoring steps.

During practice problems, the Profile model in IMTS evaluates the student's diagnostic performance, and it offers assistance in conducting an efficient and rational diagnostic process. Both of these support functions rely on Profile's ability to compute near-optimal testing decisions at each stage of a problem. Profile's generic strategy is to find, at each step, the test that offers the potential for revealing the most new information about the status of the system relative to the cost of obtaining the information. These are a function of the symptoms produced by all failures under consideration, the cost and reliability of each replaceable unit, and the time to replace each unit.

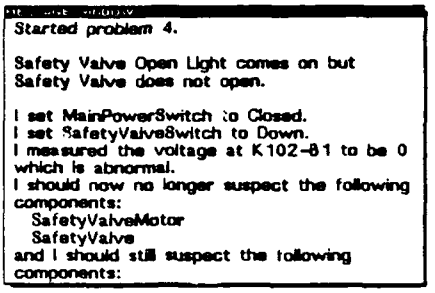
By maintaining a concurrent and internal evaluation of the symptoms seen by the student, Profile is able to evaluate each student action and to comment on its usefulness.



Profile can also generate advice tailored to the user's personal progress in working on this problem. The advice given takes into account what the student could have learned about the troubleshooting problem from the tests he or she conducted, the reliability of each element in a device, and the time to perform alternative tests and replacements.



When a student has finished a troubleshooting problem, IMTS can use Profile to present a step-by-step critique of the student's work. In a similar format, it can generate and explain an expert (Profile) approach to the problem, as shown below.

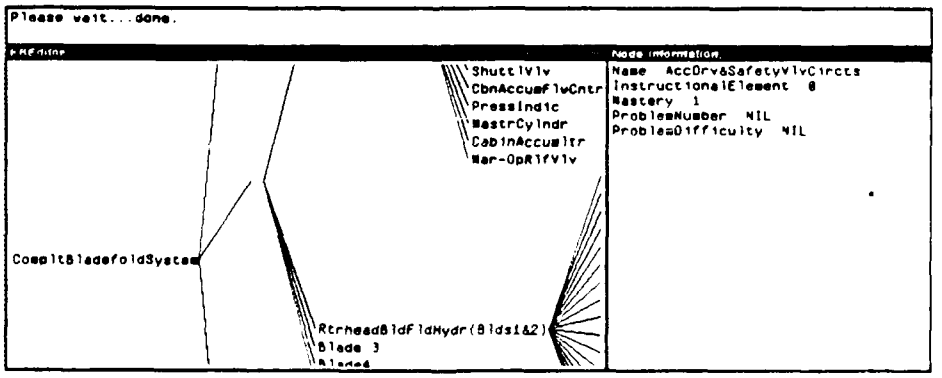


The Student Model and Problem Selection

IMTS maintains information about the student during the course of a training session. This information constitutes a simple model of the student. Three types of data about the student are maintained:

- A unitary measure of competence in the domain
- An estimate of preferred problem step size
- An overlay model of knowledge about the domain

The overlay model of student knowledge is a set of weights on the nodes of a knowledge tree that represents normative knowledge about the device that constitutes the domain of instruction. The normative knowledge model is constructed using the IMTS knowledge editor.



For each node in the knowledge tree, authors enter a number that represents an estimate of how well the average student understands the concepts it represents when they first begin working with the simulation. These estimates are called default *mastery values* for the knowledge nodes. When a new student begins working with an IMTS simulation, a copy of these mastery values is generated. This is the individual student model. These mastery values are altered in response to a student's performance, so a

student's entire set of values represents the IMTS estimate of the student's knowledge based on performance.

At the end of each problem, a mastery value is computed for the problem node. This value is based on the correctness of the problem solution, the number of errors made *en route* to the solution, and normalized time to solution. It is an estimate of the student's mastery of the knowledge required to troubleshoot the malfunction.

Mastery values are propagated upward in the tree. The immediate parent node of the problem node is modified by an amount that is proportional to the number of its children. This modified mastery value for the component node is, in turn, propagated to its parent, and so on. The solution of a single problem results in a small change even to the root node, which represents knowledge about the device as a whole.

To make an automatic problem selection, IMTS computes the *conceptual distance* from the last problem node to each of the available (not yet done) nodes. Conceptual distance from a node is the sum of the weighted links on the path between the nodes. The weights used are the inverse of the mastery values on the nodes in the paths. The automatic problem selector attempts to pick a problem with a conceptual distance that is congruent with the student's preferred problem step size. In addition, the problem selected should have a difficulty level that is congruent with the present estimate of the student's competence in the domain. These two factors — problem step size congruence and difficulty/competence congruence are heuristically combined to select an appropriate problem.

Deep Simulation

The deep simulation approach is preferred to the surface approach whenever the author has a thorough understanding of the behavior of the components of the simulated device. Deep simulations do not require the detailed authoring of effects at the device level that are required for the construction of surface simulations.

Behavior Modeled at the Element Level

The objects used in IMTS simulations can be produced by non-programmers, and they can be saved and used in any number of specific applications. This contrasts with some other approaches to simulation composition, such as that employed in STEAMER (Williams, Hollan, & Stevens, 1981; Hollan, 1983; Hollan & Hutchins, 1984) in which the simulated device is modelled with a specially written computer program. (STEAMER's graphical indicators — such as gauges and indicator lights — are generic elements that can be used at different points in a simulation, or in different simulations.) The IMTS approach has the advantage of permitting faster and easier simulation development, for the class of systems that can be simulated in this manner. The STEAMER approach has the advantage that it can simulate virtually any system, but at considerably higher cost.

One other advantage to constructing the simulation of predefined objects is that models of failed components also can be created and inserted into the simulation. This failure insertion can be done by the student, to observe and learn about effects; it can be done by the instructional routines in IMTS, to set up an instructive diagnostic problem; or it can be done by the simulator if a current mode and/or failure condition causes a new failure.

Local Propagation of Effect

Every generic object has a set of behavior rules for each of its states. One rule determines when the object will transition to the state. Other rules, called *performance effects*, determine the values of the *ports* of the object in that state. *Ports* are points on an object that are associated with the passing of values to and from other objects. In terms of the represented world, ports are electrical, hydraulic, or mechanical connections.

When a student changes the state of a simulated control object, the object's performance effect rules determine new values for some or all of its output ports. These values are passed on to the neighboring objects, some of which may change state as a result of their new input values. These objects will, in turn, pass values on to the objects to which they are connected. In a complex simulation, hundreds of objects may be affected by a single manipulation, and thousands of port values may be recomputed.

Complex system-level behaviors are derived from simpler component-level behaviors. This permits accurate free-play simulations without requiring authoring an immense number of combinatorial effects (as did some earlier simulation training systems developed by this research group, described in Towne, 1986; Towne & Munro, 1981; and Towne, Munro, Johnson & Lahey, 1983).

A major advantage of generating system behaviors from the detailed functional model is that the author is not concerned with where or when abnormal symptoms will appear in the simulated system; the effects are computed according to both connectivity and object behavior. Thus, symptoms produced by a failed part might not show up until the signal reaches a particularly sensitive indicator. It might then appear normal for a number of tests (which perhaps cannot discriminate the abnormality), followed by further abnormal symptoms.

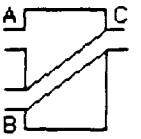
The Deep Simulation Algorithm

The simulation update is triggered when the state of an object, such as a switch, is changed. One possible effect of a state change is that an output value changes. For example, a variable voltage source would produce different values at its output port depending on the setting of the object. In this case, the port, with its new value, is put on a "source" stack. Another possible effect of a state change is that the path through the

object may be altered. For example, a valve could be changed from Straight to Crossed. In this case, each port on an altered path is put on the source stack.

The simulation driver continues running as long as there is anything on the source stack. It works by removing one item (port) from the stack and starting the propagation of that port's value through the system. Each port record stores the port's value and its connection to another object's port. The value of the first port is passed to the second port. The object containing the second port then computes what to do, given that new input value.

When a new value reaches an object, one of three things can happen:
(1) The object may serve as a dead end; the new value doesn't affect the state of the object, and there is no path through the object that includes the input port. This terminates a segment of the simulation. For example, when the valve shown at the right is in the state depicted, its port A is a dead end for values propagated to it.



(2) The value can be passed to an output port of the object. Possibly the value will be changed when it passes through the object, as in a pressure reducer; usually the value is unchanged, as in a valve, a pipe, or a wire. When the value is passed through the object, it is handed to the port of the connected object.

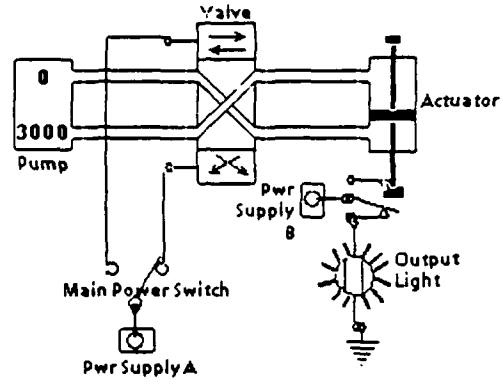
(3) The value can cause an object to change state. However, the state is not changed immediately. Instead, when it is determined that an object should change state, it is put on a "change state" stack. This stack is necessitated by the fact that an object's state may depend upon two (or more) port values; when the first value arrives (is computed), it and the old second value may dictate that the object change, but when the new second value arrives, a different result may be called for. Perhaps the object shouldn't change at all, or perhaps it should change to a different state from what was initially indicated. Consequently, no states are changed until the source stack is empty and there is no further propagation of values. At that point, an object is taken from the change state stack. If the initial instruction to change has been countermanded, then no state change is produced. If an object does change state, that will often retrigger a simulation update, since the change will usually result in one or more ports being put on the source stack.

The simulation update finally ends when there is nothing on the source stack, there is nothing on the change state stack, and no values are being propagated.

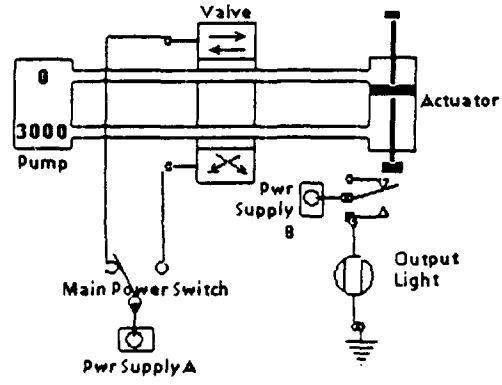
A Simple Simulation

A simulation is composed of instances of generic objects. Below is a simple simulation of a Rube Goldberg machine that uses electrical, hydraulic, and mechanical components to turn a light on and off. Power Supply A provides power through a switch to an electrically operated control valve, while Power Supply B provides power to the Output Light if the Actuator is extended.

When the user moves the Main Power Switch to the right, the valve is put in its crossed position, as shown below. This directs hydraulic pressure to the mechanical Actuator (at the right in the diagram), causing it to extend. The actuator pushes a contact closed, and electrical power turns on the Output Light.

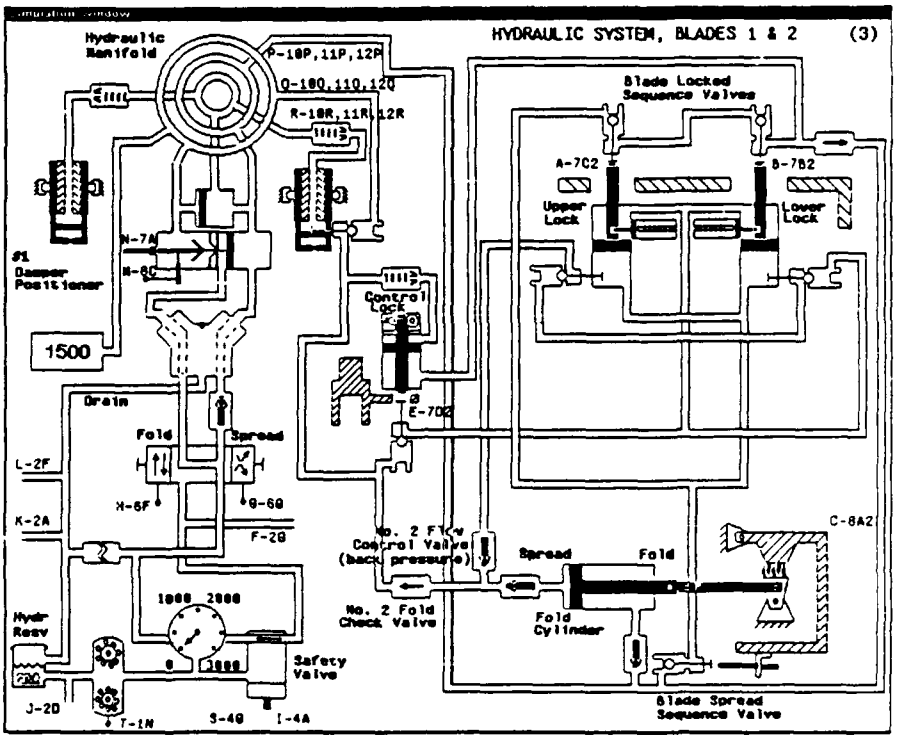


If a user moves the switch to the left, the valve goes into its straight state, as shown below, and the actuator is retracted. The contact below opens, and the Output Light goes out. All these responses are produced in accord with the behavior rules stored with each generic object.



The Bladefold Simulation

The largest deep simulation constructed thus far with IMTS is the Bladefold simulation. This is a thirteen-scene simulation with hundreds of specific objects. It simulates the blade fold/spread/brake mechanism of a military helicopter. The figure below displays one of the thirteen Bladefold scenes.



The Bladefold system uses a complex combination of electrical, mechanical, and hydraulic parts, and it is not functionally modular. A change in one component may have an effect on many other active components distributed throughout the system. In the deep simulation, therefore, a student's manipulation of a switch may result in the activations of hundreds of other objects. Despite this complexity, performance of the simulation is quite acceptable. A worst-case time to completely simulate a response to a student action is approximately 14 seconds, which is about half as long as the real-world device response. Many graphical effects are displayed during this process, so the student is actively engaged. For a more thorough description of the Bladefold Simulation, see Appendix D and F of *IMTS User's Manual* (1989).

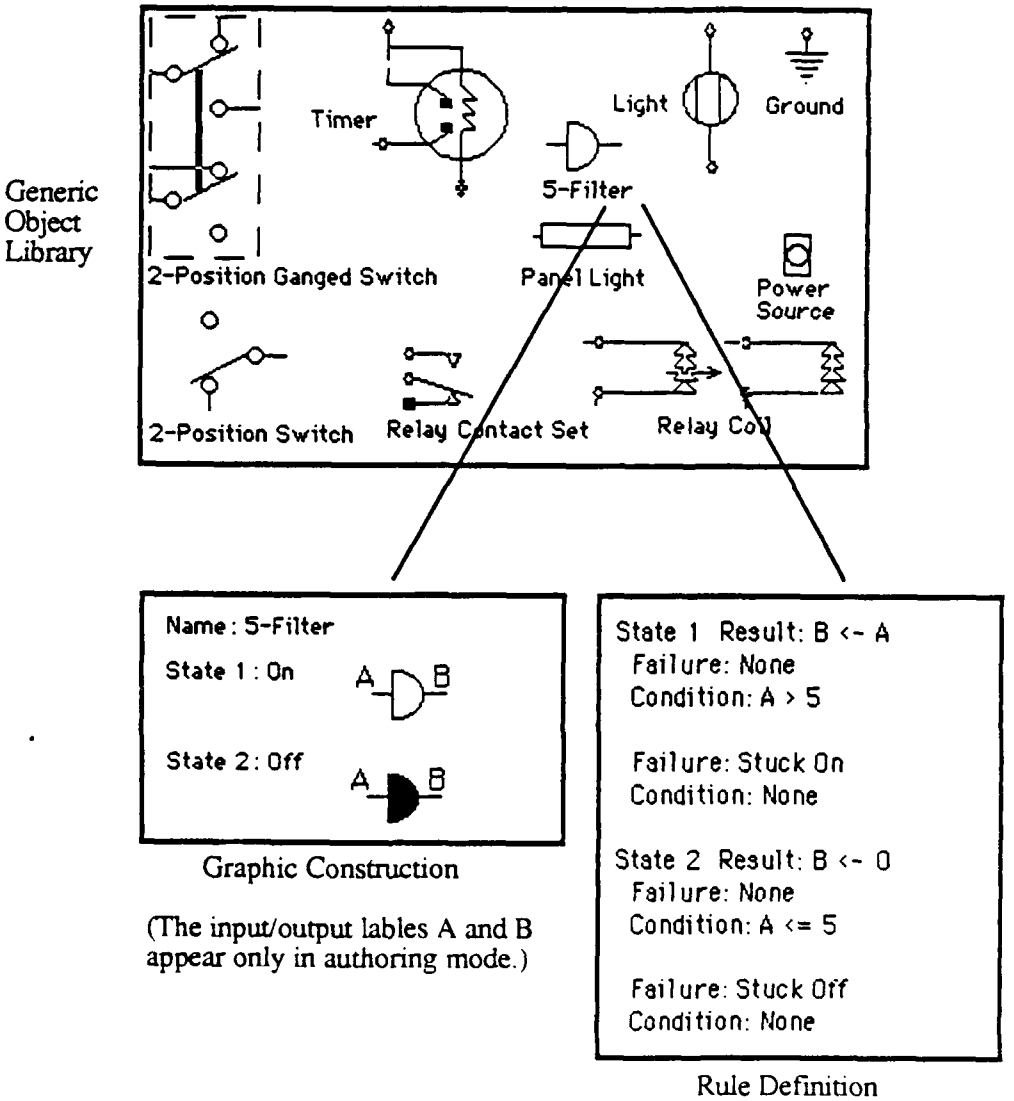
Scene Composition

IMTS simulations are composed from libraries of generic objects. As the object instances are positioned in the diagram, any input/output ports close to ports on adjacent objects are automatically noted as being connected, and can therefore exchange values.

When an individual scene of the functional model is completed the author may interact with it to verify its behaviors. The scene editor provides tools for setting input values manually, so that a scene can function independently. When all the individual scenes have been verified, the author connects them by identifying the port connections that cross scene boundaries.

Generic Object Authoring

New simulations may require some object types not available in any library of generic objects. In these cases, authors can add new generic objects to their libraries. There are two major steps to building a new generic object. First, each of the different ways that the object can appear is drawn on the screen. Second, the behaviors for the generic object are specified in terms of the *conditions* under which each state exists and the resulting *effects* produced by the object under each condition. The effects are expressed as values at output ports of the object as a function of values at input ports of the object. Thus each object is specified entirely in terms of its local environment.



Graphic Construction

(The input/output lables A and B appear only in authoring mode.)

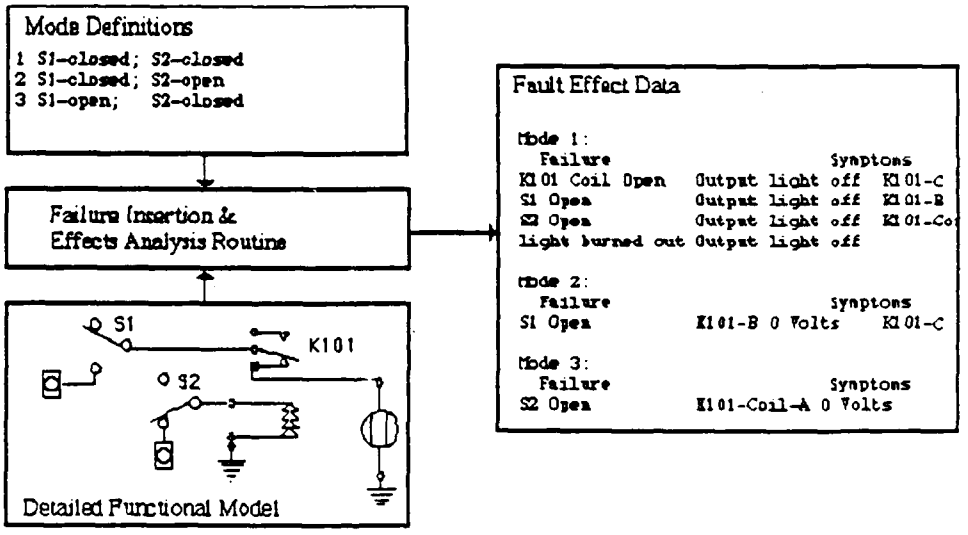
Rule Definition

A simple example is shown above. The object being added to the library, called a 5-filter, goes into an ON state (state 1) if its input at Port A is greater than 5 or if the part

has failed in a stuck-ON condition. In this state the output at port B will be the same value that is input at port A. If the input is less than or equal to 5, or if the part has failed Stuck-OFF, then the object is in the OFF state (state 2), and it outputs zero. With these behavior rules, both normal and malfunctioning situations are simulated.

Producing Fault-effect Data

Like a human troubleshooter, Profile must consider the possible effects of a vast array of failures. While the IMTS simulation reflects the effects of the current failure(s), Profile requires access to failure-effect information for all the possible failures. To minimize the compute time to generate the fault effect data, the author identifies the failures that could cause each of the *complaints* that will be used to start problems. A complaint is a verbal statement that states some abnormality that has been observed by an operator, such as *"The signal to noise ratio is low in FSK Transmit mode."* The author then executes a special batch program that automatically inserts each possible failure into the simulated device in each mode of interest, it executes the simulation to determine the effects of the fault, and it records the fault's effects. This process is illustrated below.



The resulting table of fault effect data allows Profile to rapidly search for powerful next tests, to evaluate the power of the student's test selections, and to determine and explain the significance of test result obtained by the student, as he or she interacts with the IMTS simulation. Profile conducts its analysis by maintaining a set of suspicion levels for possible malfunctions. Initial suspicion levels are determined from the inherent reliabilities of the parts of the device. These could be modified over time to reflect field experience with a particular device. As students perform tests in IMTS, Profile modifies the current suspicion levels to reflect the inferences that could be drawn from the results.

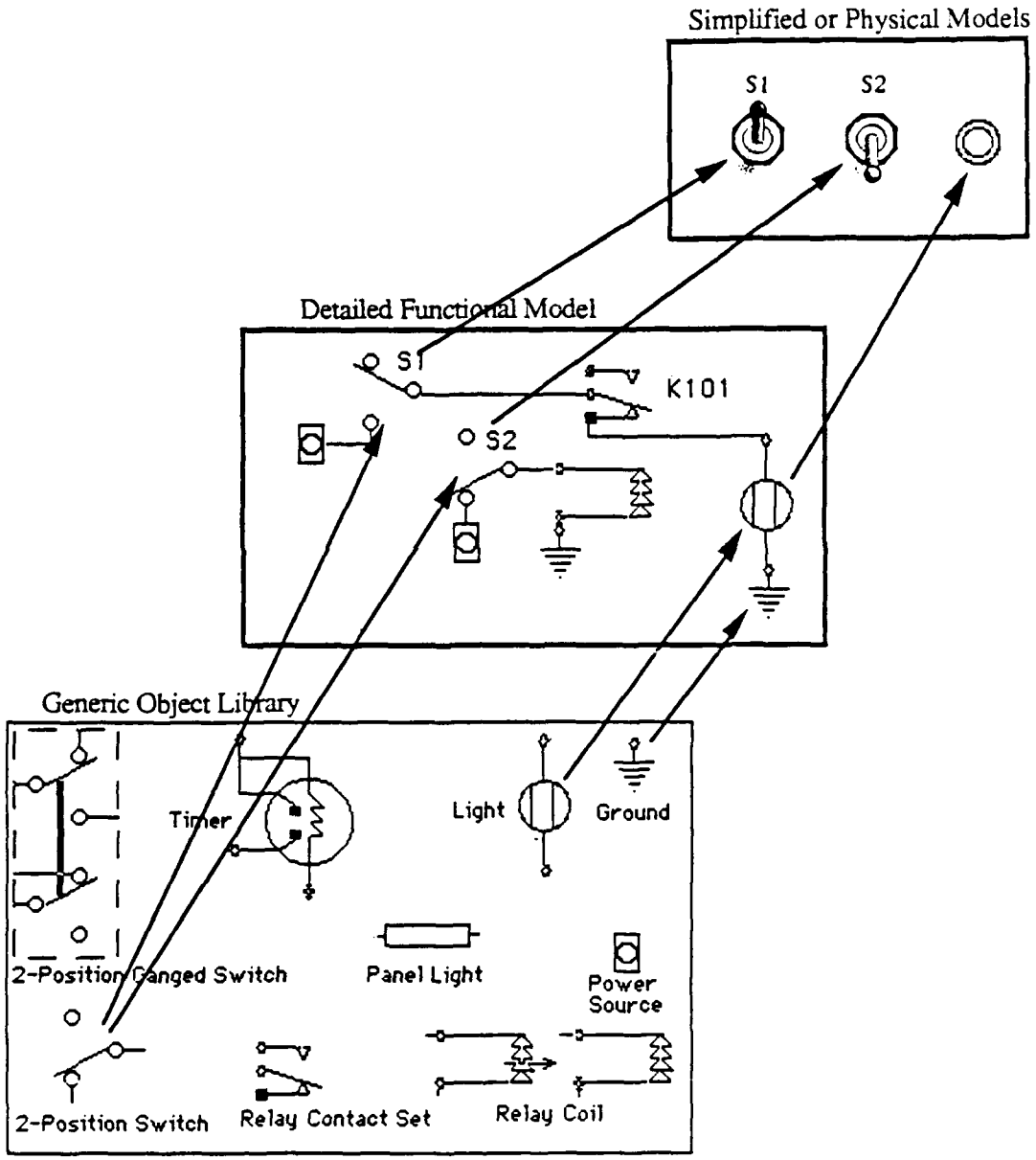
Development and Applications of Derivative Simulations

For many devices a complete IMTS simulation will be too complicated to be used by introductory students. IMTS provides a mechanism for creating simplified simulations derived from the detailed functional model. These simplified simulations operate correctly, even though some or many of their critical parts are not shown, because the parts that are shown obtain their behaviors from their counterparts in the complete functional model. This allows the simplified models to appear to operate correctly, even though they could not really operate in the real-world without the missing parts.

The IMTS authoring feature that supports these derivative simulations is called *yoking*. One object can be yoked to another, meaning that the behavior of the yoked object will be determined, not by the behavior rules of its own generic type, but rather by the behavior of the specific object to which it is yoked.

Yoking can also be used to rapidly create physical representations of systems. These representations may be appropriate for training device operation as well as fault diagnosis using front panel indicators. In the figure on the next page, a detailed functional model has been developed from the IMTS Generic Object Library. Then a three-element scene was produced by yoking S1, S2, and the indicator light to their schematic counterparts in the Detailed Functional Model (the detailed IMTS simulation model). If a student sets S1 on the front panel view, the indicator light on the front panel view will come on because S1 was automatically set in the functional model, and the indicator light there was determined to be in the ON state.

When IMTS simulations are used to support training in equipment operation, this more physical form is usually desired, although the underlying schematic form may be a powerful explanatory tool.


Simplified or Physical Models
S1
S2
Detailed Functional Model
S1
K101
S2
Generic Object Library
Timer
Light
Ground
2-Position Ganged Switch
Panel Light
Power Source
2-Position Switch
Relay Contact Set
Relay Coil

Surface Simulations

Surface simulations are developed by explicitly specifying behaviors of objects in relation to other objects in the particular device. Authors use the surface simulation methodology for devices that are too complex to describe in terms of the independent behaviors of their components.

Previous Surface Simulation Systems

The surface simulation methodology in IMTS is closely akin to that used in some earlier training systems developed at Behavioral Technology Laboratories. The technique provides a way to specify complex system behaviors in terms of specific conditions rather than by enumerating the countless combinations of switch settings and failure states. The conditions express the settings of switches and the existence or absence of failures.

While the surface simulation approach is considerably less robust than the deep simulation approach, it does provide a way to simulate systems whose inner workings are either not fully understood by the simulation developer or are too complicated to handle. Thus, a technician having extensive field experience with a system could produce a fully accurate surface simulation by working out all the various effects of front panel configurations and malfunctions on indicators and test points, even though that individual might not fully understand the functions of the individual units.

The past limitations of surface simulation have been significantly overcome in IMTS. The most clumsy aspect of the Generalized Maintenance Training System (GMTS) and EEMT, its commercially produced version, was the medium employed to simulate the device. For GMTS this medium was randomly retrievable color microfiche images, for EEMT the medium was videodisc. While the videodisc version retrieved and displayed images quickly and reliably, it suffered from the same limitation as the microfiche version, namely that fixed photographic images were required of every system state in the simulation.

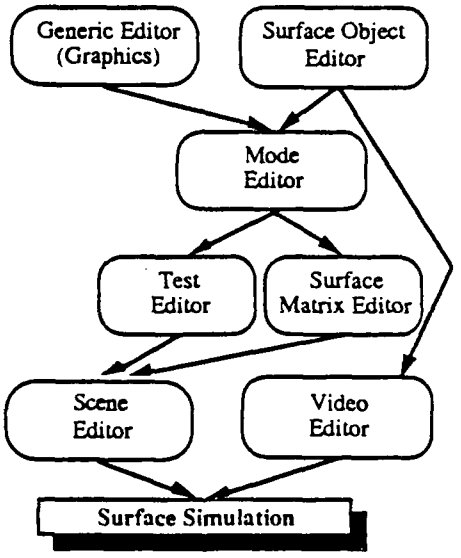
Because the number of combinations of switch settings and indicator readings was astronomical, the only recourse in these earlier systems was to minimize the contents of each scene. Thus a typical scene might contain four to six switches and a few indicators, and would require between sixty to two hundred different photographs to reflect all the different combinations of object states. Not only was the time and cost to produce these images substantial, but the limited scene size worked in direct opposition to the desired goal of realistically simulating the real device.

In IMTS, the graphic representation of each object is produced independently, thus scenes can be as large as the display screen can accommodate, and the viewer of a simulation cannot detect whether it was produced using surface or deep techniques.

The second limitation to earlier surface simulations was that they embraced no instructional or diagnostic expertise. The instructional and diagnostic functions in IMTS operate for surface simulations as well as deep simulations. The difference here is that the fault-effect information for surface simulations must be supplied by a human expert, whereas it is generated automatically for deep simulations.

Overview of Surface Simulation Authoring

A surface simulation author constructs scenes of graphical objects that are yoked to the surface objects. These graphical objects get their appearances from libraries of generic objects, just as do the objects of deep simulations. Their behaviors, however, are not based on the behavior definitions in a library. Instead, their behaviors — that is, their changes in appearance — are driven by current state data associated with surface object data records to which they are yoked.



Four editors are used to define surface simulation behaviors. The *Surface Object Editor* creates specific objects that have certain data associated with them, such as their names and types. These specific objects need not be associated with any graphical objects. The *Mode Editor* is used to define equipment modes of interest in terms of the states of specific objects. The *Test Editor* defines tests, which are indicators and/or test points viewed in particular modes. The *Surface Matrix Editor* is used to declare what values a test should exhibit in various failure conditions.

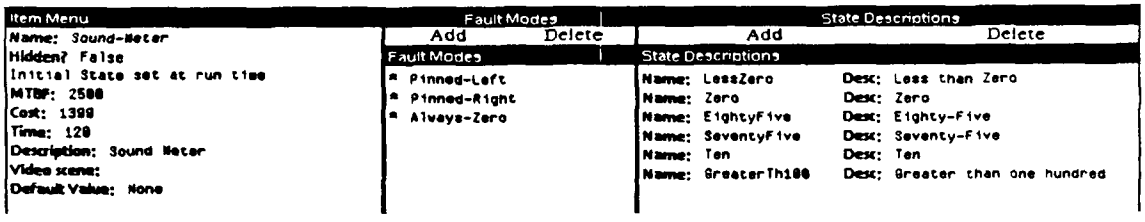
Surface Object Data

For every switch, indicator, jumper, or other replaceable object in a device, the author enters a data record describing that object. These data records are called *surface objects* in IMTS. The behavior of a surface simulation is determined by these data records and by others that describe the important modes of the device and tests that can be

performed in those modes. Surface objects, *per se*, do not have appearances; they are only bundles of data, including the following attributes:

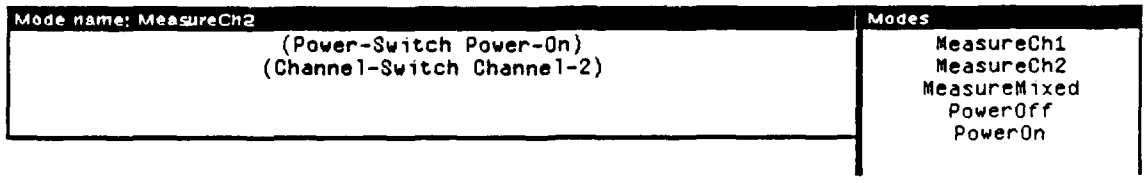
- Name
- Hidden status (the default is False)
- Initial state (set at run time for test points & indicators)
- MTBF, cost, replacement time (for Profile diagnostic guidance)
- Description (used for discussing with the student)
- Video scene (the name of the video scene on which the object appears)
- Default Value

The *surface object editor* , shown below, is used to enter such data for a simulation.

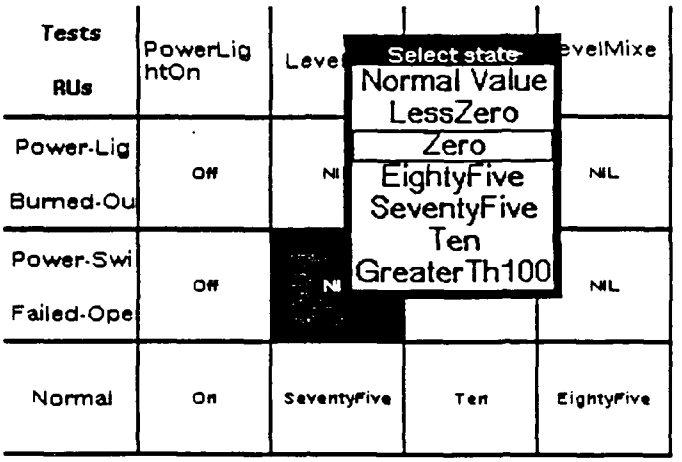


Modes and Tests

Authors define the *modes* of a surface simulation. A *mode* is a combination of object states of interest. For example, the power switch being set to *on* is a simple mode of interest. A more complex mode is power *on* and standby *off* and number 1 engine switch set to *start*. Mode information is edited with the *mode editor*.



A *surface matrix editor* is used to specify how indicators and test points behave based on the defined modes in a number of malfunction states. For each mode that is relevant for an indicator, the value or state of the indicator is specified for each malfunction.

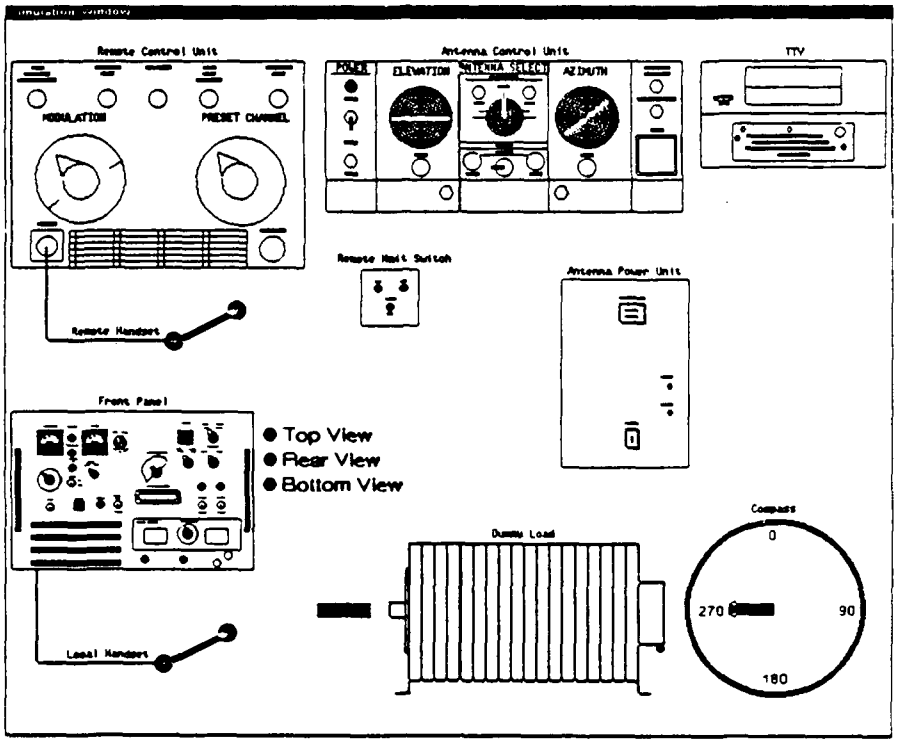


This fault-effect matrix is identical to that used for deep simulation. The only difference is that in surface simulations the symptom data must be supplied by a human expert.

During a simulation, when the student changes a switch, all the modes that refer to that switch check to see whether their truth has changed. Certain modes may become true as a result of the switch manipulation. For each mode that changes, all of the indicators that refer to that mode are updated. If an indicator's state changes, then the graphic object that represents that indicator is redisplayed in the new state.

The WSC-3 Simulation

The largest surface simulation constructed thus far with IMTS is the AN/WSC-3 Simulation. The AN/WSC-3 is a satellite communication system used for voice and data communication. The WSC simulation, comprised of twenty-eight scenes, demonstrates the surface simulation capabilities of IMTS. The figure below shows the top-level scene of the system.



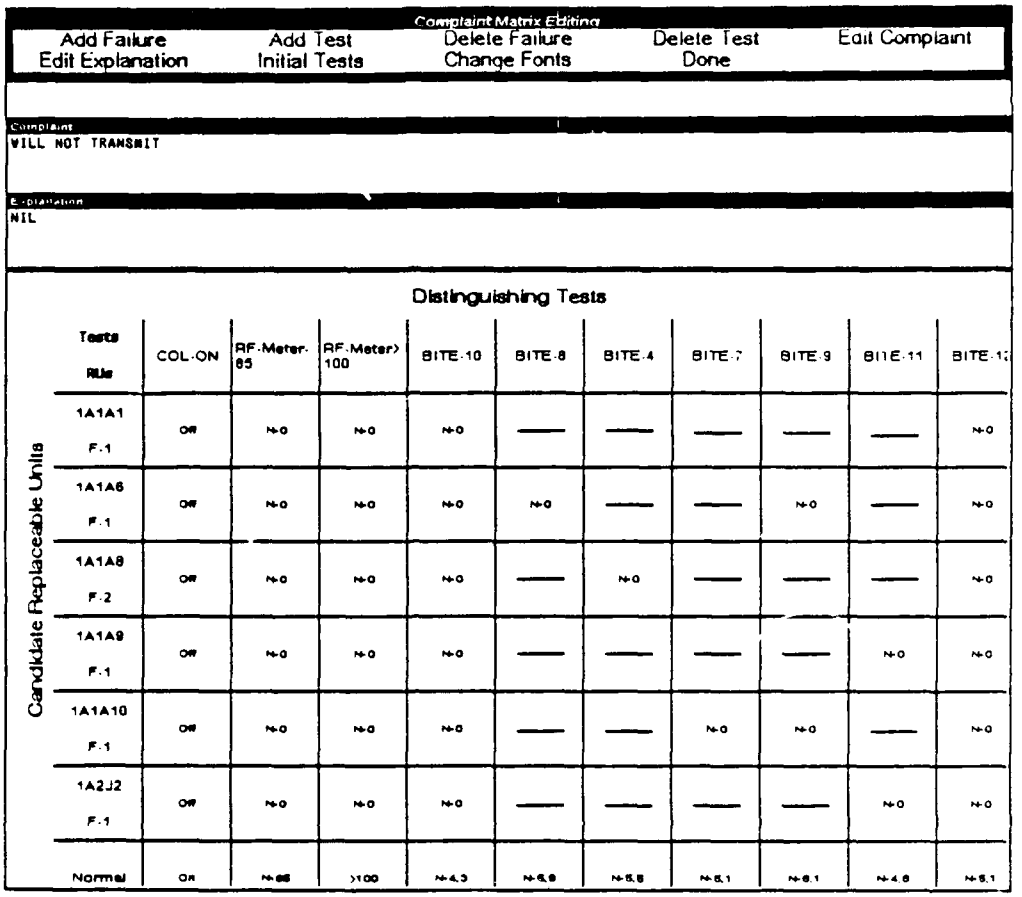
A WSC-3 Simulation Scene

Most of the graphic objects in the top-level scene are icons that lead to more detailed views. When the student clicks on the Front Panel unit, for example, IMTS displays a close-up view, showing the current switch settings.

While videodisc is not a convenient medium with which to represent panels of equipments in each of the possible modes, it is quite attractive as a means to display test equipment readings. When the student performs an oscilloscope reading on the WSC-3, he or she sees the waveform as a photographic image retrieved from the videodisc.

Creating Profile Data for Surface Simulations

After defining the normal behavior of a surface simulation using the test editor, the author prescribes how the device behaves in various malfunction states. The *surface matrix editor* is used to describe the behavior of the simulated device under a given complaint. A *complaint* is a statement of abnormal behavior that applies to a number of related failures. The matrix below specifies the behavior of the WSC-3 system for the malfunctions that prevent normal transmission.



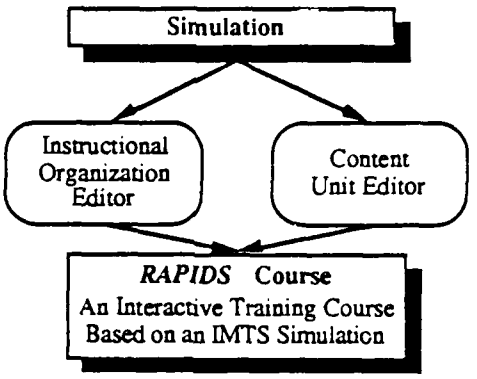
In surface simulations this table of data supports the simulation as well as supplying Profile with the necessary symptom information.

Authoring Instruction by Direct Manipulation

Until recently the instructional applications of IMTS have been limited to intelligent support of practice in fault diagnosis. Because IMTS generates all instructive interactions, it cannot offer explicit instruction in such topics as theory of operation, front panel configurations, symptom identification, or test interpretation (although IMTS practice involves these skill components).

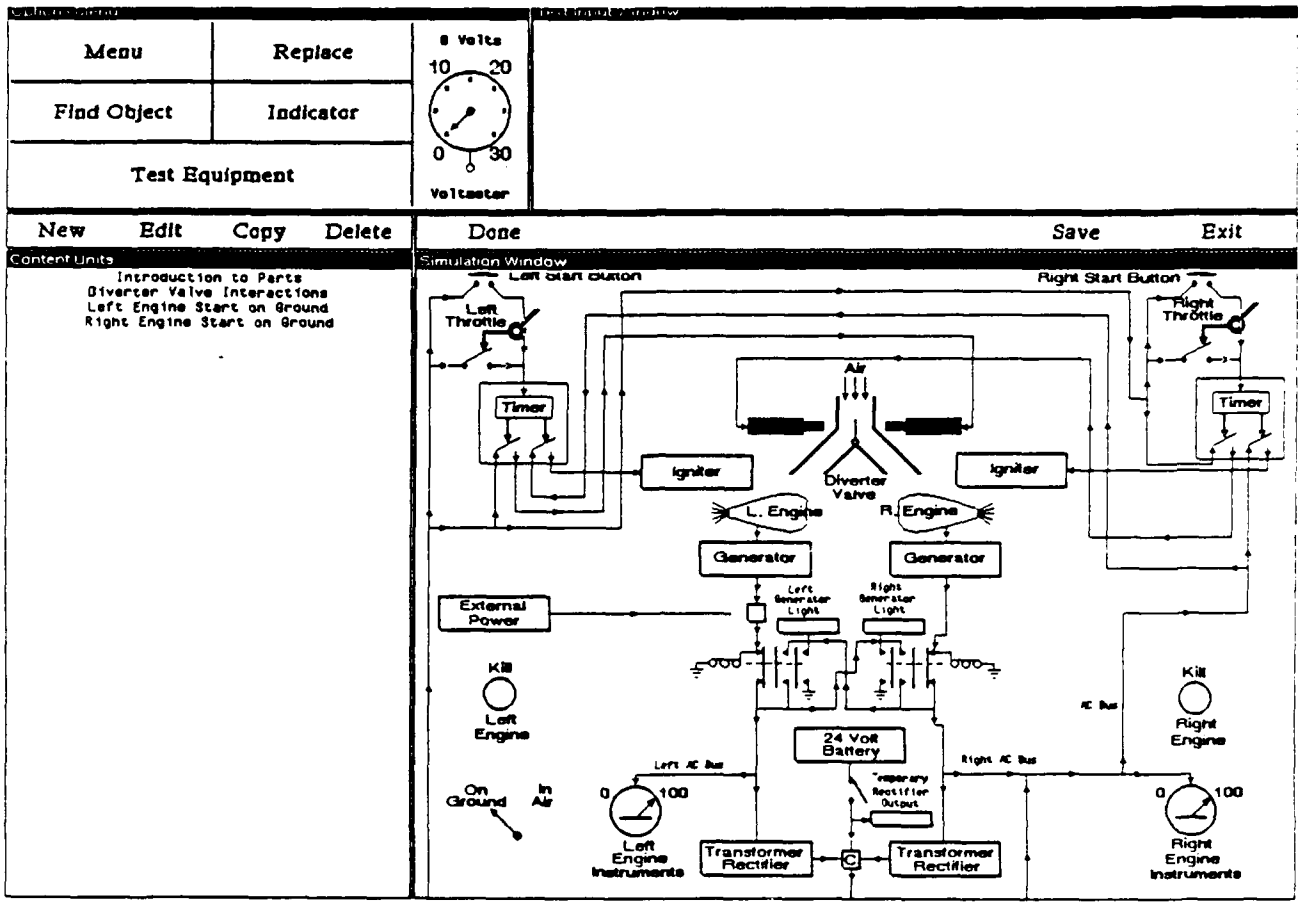
RAPIDS (Rapid Prototype ITS Development System) was developed to provide additional features for authoring and delivering instructional interactions, based on IMTS simulations. Using the RAPIDS tools, authors can create instructional units on almost any device-related topic, largely by performing the procedures that they want their students to learn on the simulation. RAPIDS is extensively described in Towne & Munro (1989) and in Munro & Towne (1989).

After creating an IMTS simulation, authors can build domain-specific content units, using a *content unit editor*. These content units are organized into an instructional plan using an *instructional organization editor*. Authoring is direct and largely error-free because it is built on the foundation of an IMTS simulation.



Instructional Content Authoring

RAPIDS instruction is created by operating a live IMTS simulation, and by adding text and graphics to highlight and explain the procedures and effects being demonstrated. The figure below shows the RAPIDS content editor being used to create instruction for an aircraft engine starting system (adapted from Kieras, 1988).



Authoring Content Units

A *content unit* is authored by performing operations on the simulation and by creating instructive *expositions*, before and/or after each action, that explain and elaborate on what the expert is doing and how the device is responding. The action might change the state of the simulated device, such as setting a switch or replacing a (simulated) defective part. It might reveal something about the state of the simulated device, such as making a test reading using simulated test equipment. Or, it might be a simple identification of some object or area in the simulation graphics, or a response on a multiple-choice list.

The expositions can highlight portions of the simulation display, they can play videodisc frames, they can display text, and they can control waiting for various events or time durations. The text in expositions may be presented in a standard text window at the side of the simulation graphics or it may be positioned on the graphic simulation to relate closely to particular parts of the device representation. The author has the option of setting the simulated device into a particular mode of operation prior to the unit.

Typically, the unit is first played for the student in an *instruct* mode. Each of the author's actions are automatically performed along with the accompanying text and/or videodisc expositions. In this mode the student simply studies the simulated actions, the device responses, and the accompanying explanations, and paces the presentation according to his or her own learning speed. Then the unit can be presented in drill mode. As in instruct mode, the learner sees the instructive expositions, but now attempts to perform all the actions and selections. All errors are automatically corrected by RAPIDS.

Depending upon the course plan, the same unit presented in instruct and drill mode may also be presented in test mode. In this mode the learner does not see the instructive expositions, and attempts to perform the procedures and drills unaided. Throughout the presentations, RAPIDS automatically monitors and remediates the learner, and it maintains performance scores for each learner on each unit.

Student Actions

The specification of a student action may be any of the following:

- selecting or identifying one or more objects or areas on the simulation
- manipulating one or more switches into specified states
- replacing a simulated object
- performing a specified test using simulated test equipment
- making one or more selections from a menu of text items

The last of these options provides a mechanism for specifying multiple choice questions and answers. A simple user interface provides a straightforward implementation that does not require any special authoring techniques.

Expositions

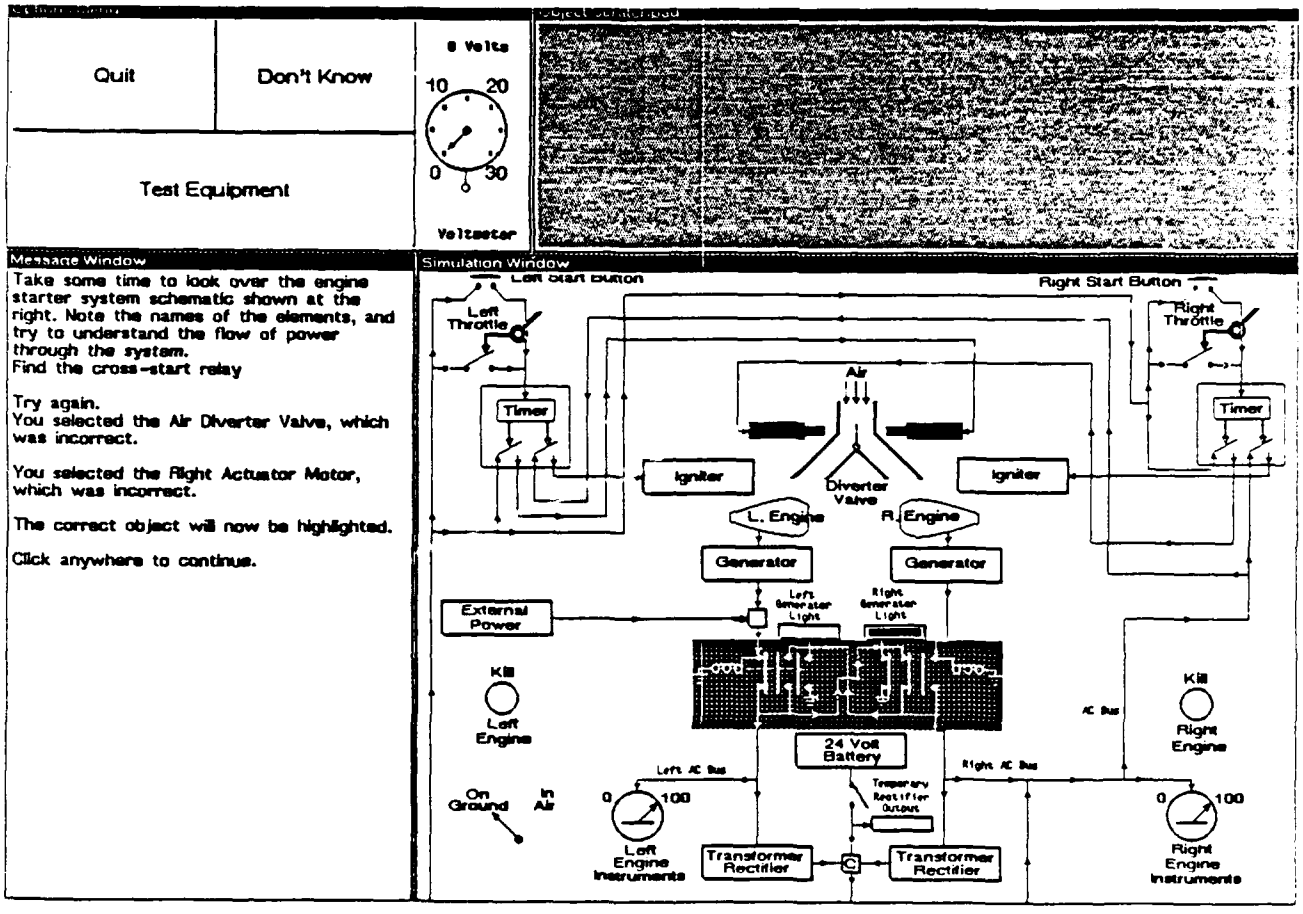
An exposition may consists of a combination of the following exposition elements types:

- presenting text in the message window or a floating window
- clearing the message window
- playing a videodisc segment
- highlighting an object or region in the simulation window
- unhighlighting an object or region in the simulation window
- changing the scene displayed
- waiting for a student response
- waiting for a specified amount of time

Automatic Interactions with Students

The built-in functions in RAPIDS automatically generate many standard student interactions. These include providing informative feedback on errors, giving help on request, evaluating performance, and repeating content units as required.

In the figure below, a learner was unable to locate the Air Diverter Valve. RAPIDS attempted to resolve the confusion by informing the learner what he had mistakenly taken for the Air Diverter Valve. After two errors, RAPIDS showed the learner the correct answer.

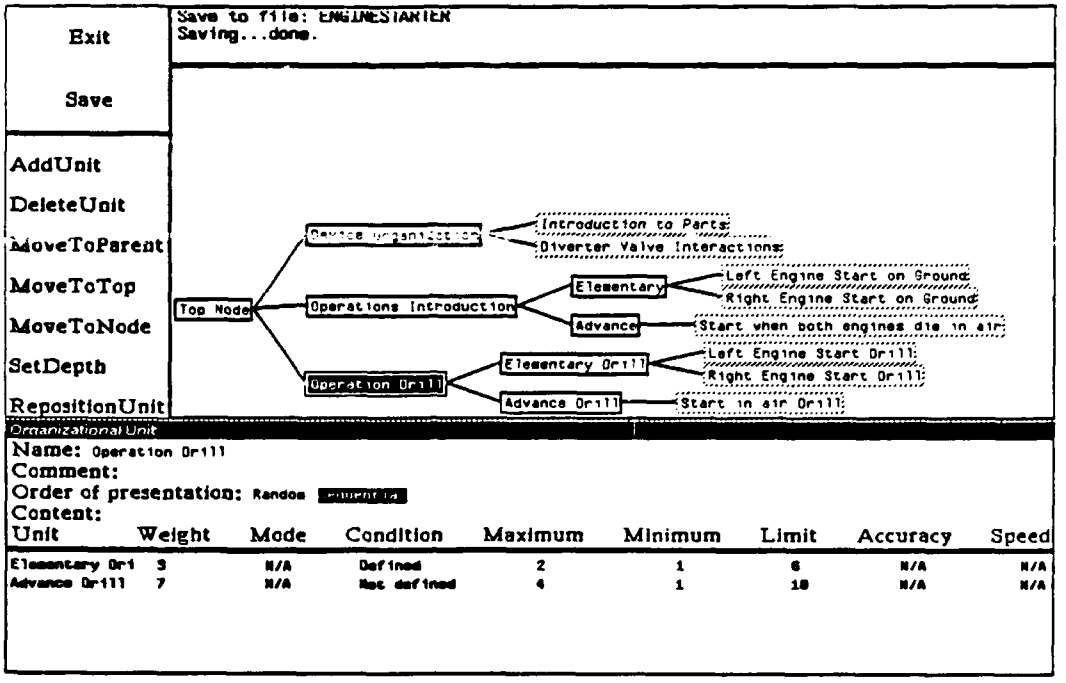


Instructional Organization

RAPIDS provides an interactive tool for creating and editing instructional plans for courses. The instructional plan specifies what content units will be presented, in what mode (instruct, drill, test) they will be presented, and under what conditions they will be presented to individual students. The plan also specifies how much time can be devoted to the various units, how many times a unit may be repeated, and what speed and accuracy scores are required to complete a unit.

The window below shows a plan for a simple course about an engine starter system. Plans are organized as tree structures of blocks. The terminal blocks, shown in dashed lines, are content units that are individually produced using the content unit editor described above. The blocks shown in solid lines are called *organizational units*. These

are simply groups of other units. The course editor is used to add, delete, and move units, and to specify the manner in which the units will be delivered to the learner.



The instructional plan shown above organizes the course into three major topics. Students will first learn about the device organization, then be introduced to operations, and finally be drilled on operations. These three topics are each structured as organizational units. The unit covering Device Organization consists of two content units, each of which includes some subject matter to be delivered, whereas the other two major topics consist of further sub-topics, or organizational units.

The window below the tree window displays data about the currently selected unit. The data can be edited in this window. In this example, the organizational unit called 'Operation Drill' has been selected. The parameters shown with each sub-topic specify the manner in which the unit will be instructed.

Structure of Instructional Plan Units

An *organizational unit* lists other units to be presented. The member units may be content units or other organizational units. Associated with each unit in a course plan are these data fields:

- weight: the importance of the called unit (relative to the others in the list)
- mode: whether to execute a called content unit in Instruct, Drill, or Test mode
- condition: an optional expression that controls whether to present the unit to a learner

- maximum: the maximum number of times to present the unit
- minimum: the minimum number of times to present the unit
- limit: the time limit for the unit, in minutes
- accuracy: the accuracy score (%) required to complete the content unit successfully
- speed: the speed score required to complete the content unit successfully, in minutes

RAPIDS automatically computes a composite accuracy score at all levels of the instructional plan. The *weight* parameter listed above is used to compute this figure. RAPIDS also maintains speed scores for all units, reflecting the time spent by each learner at each level.

The *condition* parameter is an option that specifies the conditions under which a unit will be presented. The condition can be any expression in terms of the performance of the learner on any unit, the time spent in various units of instruction, and the number of repetitions of units by the learner.

Thus a single instructional plan can deliver quite different courses to different learners depending upon the performance of each. The course content, time devoted to topics, repetitions of topics, and degree and type of remediation are all tailored by RAPIDS to meet the high-level specifications of the course plan while recognizing the details of individual student performance.

Summary

The tools in IMTS and RAPIDS provide twelve different modules for constructing simulation based instruction of one type or another, and three interactive instructional delivery programs. The author uses the menu shown below to select and execute the module to be used. The first column contains the two key editors for building graphical simulations, the Generic (object) Editor, and the Scene Editor. The former of these is run to add new objects to the Object Library. After creating one or more scenes with the Scene Editor, the author selects Build Simulation, which compiles the newly constructed simulation scenes for execution. The simulation may then be started by selecting Run Simulation from the menu.

RAPIDS Tools			
Simulation	Surface Model	Diagnostic Training	RAPIDS
Generic Editor	Object Editor	Deep Matrix Editor	Plan Editor
Scene Editor	Mode Editor	Problem Editor	Content Editor
Build Simulation	Test Editor	Run IMTS	Run RAPIDS
Run Simulation	Matrix Editor		
	Video Editor		

Alternatively, surface simulations are constructed using the five editors listed in the second column. These editors accept information about surface objects, equipment modes, tests, fault effect data, and videodisc frame numbers.

Two editors are used to produce the fault effect data needed to support diagnostic training for a simulation, as listed in the third column. The Deep Matrix Editor accepts specifications of modes and failures and then executes the batch program that generates the fault effect data required by Profile. The Problem Editor accepts problem specifications (failed objects and failure modes) and associated verbal statements (complaints) that are presented at the start of each troubleshooting problem. After doing these two steps a simulation can be run in the IMTS intelligent training mode by selecting Run IMTS.

After creating a simulation, RAPIDS courses are built using the two editors listed in the fourth column. The author uses the Plan Editor to construct and modify the instructional plan, and the Content Editor to perform and explain tasks on the simulation. Typically, an author would construct a simulation using the editors in either the first or second column (deep or surface), then use the RAPIDS tools to create a complete course. The diagnostic training functions would be utilized only if IMTS (Profile-guided) diagnostic problems are also to be presented.

Conclusions

IMTS was developed to bring together and apply a number of diverse and experimental techniques in intelligent tutoring. The two most fundamental concepts that influenced the design were 1) the use of a responsive, object-oriented graphical model that could be manipulated by a learner, as pioneered in the STEAMER project, and 2) the generation of tailored instructive interactions from generic models of diagnostic and instructional expertise.

In addition to the two major applications developed under contract, several dozen small applications have been created by others in two workshops conducted at BTL. The participants in these workshops were provided three days of training in producing IMTS simulations, and then devoted one to two days developing small applications.

These test applications revealed numerous areas where either the authoring facilities or the user documentation could be improved or corrected. All errors were corrected, and wherever possible, features that simplified the authoring task were implemented. Screen images from many of these projects are included in the appendix to this report.

The workshop participants included technical subject matter experts and instructional researchers, having computer programming skills ranging from none to proficient. Their performance indicated that developers with a relatively wide range of skills could become productive IMTS users fairly quickly, and that they could apply the system effectively. Finally, the range of simulations that were produced provided a good indication of the generality of the techniques.

Latest Developments

The most recent work on IMTS, as described in this report, was conducted 1) to expand the applicability of the technique to a considerably wider range of devices; 2) to provide tools needed by a much broader community of instructional developers; and 3) to produce instruction for a much broader range of proficiency levels.

Extended Device Applicability

The incorporation of the surface-level simulation technique into IMTS provides developers a way to utilize the resources of IMTS even when the device to be instructed cannot be reasonably represented at the object level. While the basic surface level simulation approach employed in IMTS is very similar to that used in earlier systems (GMTS, EEMT, and ESAS), the ease of development is vastly improved in IMTS, owing to the ability to specify and depict objects individually rather than collectively. Thus the development of economical high-resolution computer graphics has had profound impact on the internal representations of devices as well as their external manifestations.

The simulation of the WSC-3 Satellite Communications System was produced in a very short time, owing primarily to the previous experience of the two subject matter experts involved. One of these had extensive experience operating and maintaining the WSC-3 system; the other had previously produced a GMTS data base for it. While this application therefore does not provide a representative test of development effort, it does indicate the speed with which a surface simulation is produced, given that the subject matter knowledge is extensive.

Extended Range of Instructional Resources and Strategies

The demonstrated instructive potential of IMTS attracted numerous potential developers who wished to utilize the tools, but who wanted to apply different or more varied instructional strategies and scenarios than those built in to IMTS. The interactive authoring facilities developed in RAPIDS allows diagnostic training to go beyond the

four scenarios of initial IMTS (practice problems, expert demonstration, problem debriefing, and free exploration).

Using RAPIDS, one can develop drills for familiarizing students with terminology and topology, exercises in associating symptoms with possible causes, and instructional units presenting theory of operation and troubleshooting. Beyond diagnosis, RAPIDS can be used to develop courses in device operation, preventative maintenance, or safety procedures. Moreover, the course developer can control allocation of time and the criteria for meeting instructional objectives.

Extended Range of Learners

The original IMTS is an effective tool for sharpening the diagnostic skills of intermediate to expert troubleshooters on a particular device. Having absorbed much of the theory of operation via conventional lectures, such technical students can gain much from working practice problems with IMTS support. The entering student, however, could not realistically attempt even the easier problems in a device as complex as Bladefold because the IMTS representation of the device was necessarily complete. While a developer might be able to produce simpler models for the novice student, doing so would necessarily entail producing many new and artificial objects, whose behaviors are also simplified, just for the purpose of supporting simpler representations.

The derived simulation capabilities described in this report allow developers to produce a series of successively more complex and complete representations of a device. Thus simple models of complex systems function as they should even though critical parts are absent from the simplified representation. Now IMTS can be applied in a manner that is consistent with the instructional philosophy of White and Frederiksen (1987), a philosophy of successive model elaboration to which we heartily subscribe.

The derived simulation features also allow development of device representations that are physically realistic, thereby providing an appropriate interface with which to practice device operation and other procedures, as well as use of front panels for diagnostic functions.

Maintaining Cognitive Fidelity

While our goals have included allowing the learner to experience simpler worlds before more complex ones, we have also attempted to maintain those cognitive aspects of fault diagnosis that lie at the heart of this difficult process, whatever the difficulty of the problem being presented. Consequently IMTS provides the learner with practice in performing tests as well as selecting them, in interpreting tests as well as observing them, and in making inferences about possible causes that require understanding component behavior as well as system structure. With the addition of the RAPIDS authoring features, part-task drills can be developed that instruct separable cognitive skills in equipment operation as well as diagnosis.

References

Hollan, J. D. 1983. STEAMER: An Overview with Implications for AI Applications in Other Domains. Presented at the Joint Services Workshop on Artificial Intelligence in Maintenance, Institute of Cognitive Science, Boulder, CO: October 4-6, 1983.

Hollan, J. D., Hutchins, E. L., and Weitzman, L. 1984. STEAMER: An Interactive Inspectable Simulation-based Training System, *The AI Magazine*, 1984, 2.

Kieras, D.E. What mental model should be taught: Choosing instructional content for complex engineered systems. In J. Psotka, L.D. Massey & S. Mutter (Eds.) *Intelligent Tutoring Systems: Lessons Learned.* 1988, Hillsdale, NJ: Lawrence Erlbaum Associates.

Munro, A. and Towne, D. M. *RAPIDS User Manual.* Los Angeles: Behavioral Technology Laboratories, University of Southern California, August 1989.

Norman, D. A. and Draper S. W. (Eds.) *User-centered system design: New perspectives on human-computer interaction..* Hillsdale, NJ: Lawrence Erlbaum Associates, 1986.

Towne, D. M. A generalized model of fault-isolation performance. *Proceedings, Artificial Intelligence in Maintenance: Joint Services Workshop*, 1984.

Towne, D. M. A generic expert diagnostician. In *The Proceedings of the Air Force Workshop on Artificial Intelligence Applications for Integrated Diagnostics*, 1986.

Towne, D. M. The generalized maintenance trainer: Evolution and revolution. In W. B. Rouse (Ed.), *Advances in man-machine systems research, Vol 3*, JAI Press, 1986.

Towne, D. M. and Munro, A. *Generalized maintenance trainer simulator: Development of hardware and software.* (Technical Report No. 81-9) San Diego: Navy Personnel Research and Development Center, 1981.

Towne, D. M. and Munro, A. *Preliminary design of the advanced ESAS system.* (Technical Report No. 105) Los Angeles: Behavioral Technology Laboratories, University of Southern California, December 1984.

Towne, D. M. and Munro, A. *RAPIDS: a simulation-based instructional authoring system for technical training* (Technical Report No. 112). Los Angeles: Behavioral Technology Laboratories, University of Southern California, September 1989.

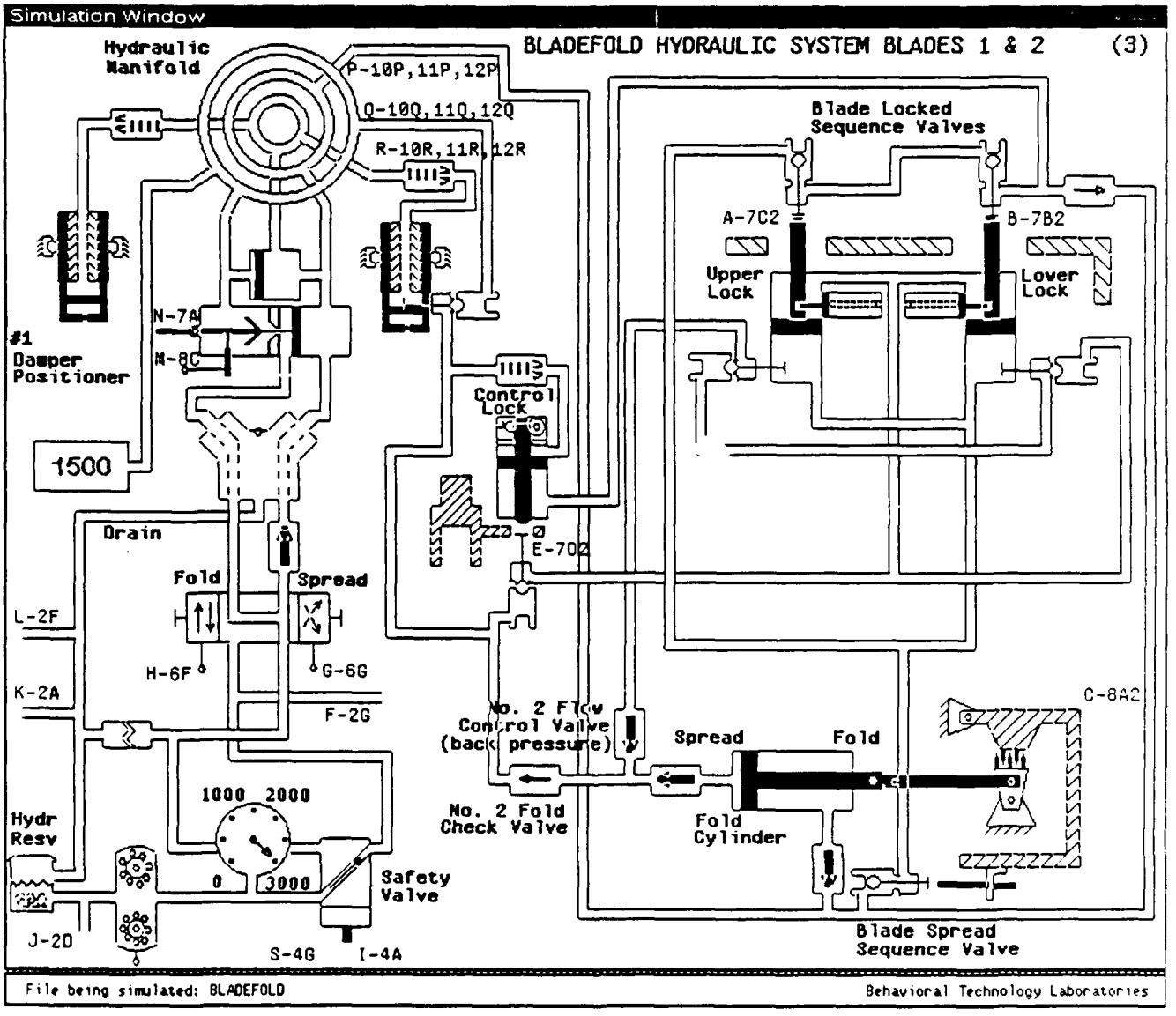
Towne, D. M., Munro, A., Johnson, M. C., and Lahey, G. F. *Generalized maintenance trainer simulator: test and evaluation in the laboratory environment.* (NPRDC TR 83-28) San Diego: Navy Personnel Research and Development Center, August 1983.

White, B.Y., and Frederiksen, J.R. Qualitative models and intelligent learning environments. In R. Lawler & M. Yazdani (Eds.), *AI and education.* Norwood, NJ: Ablex, 1987.

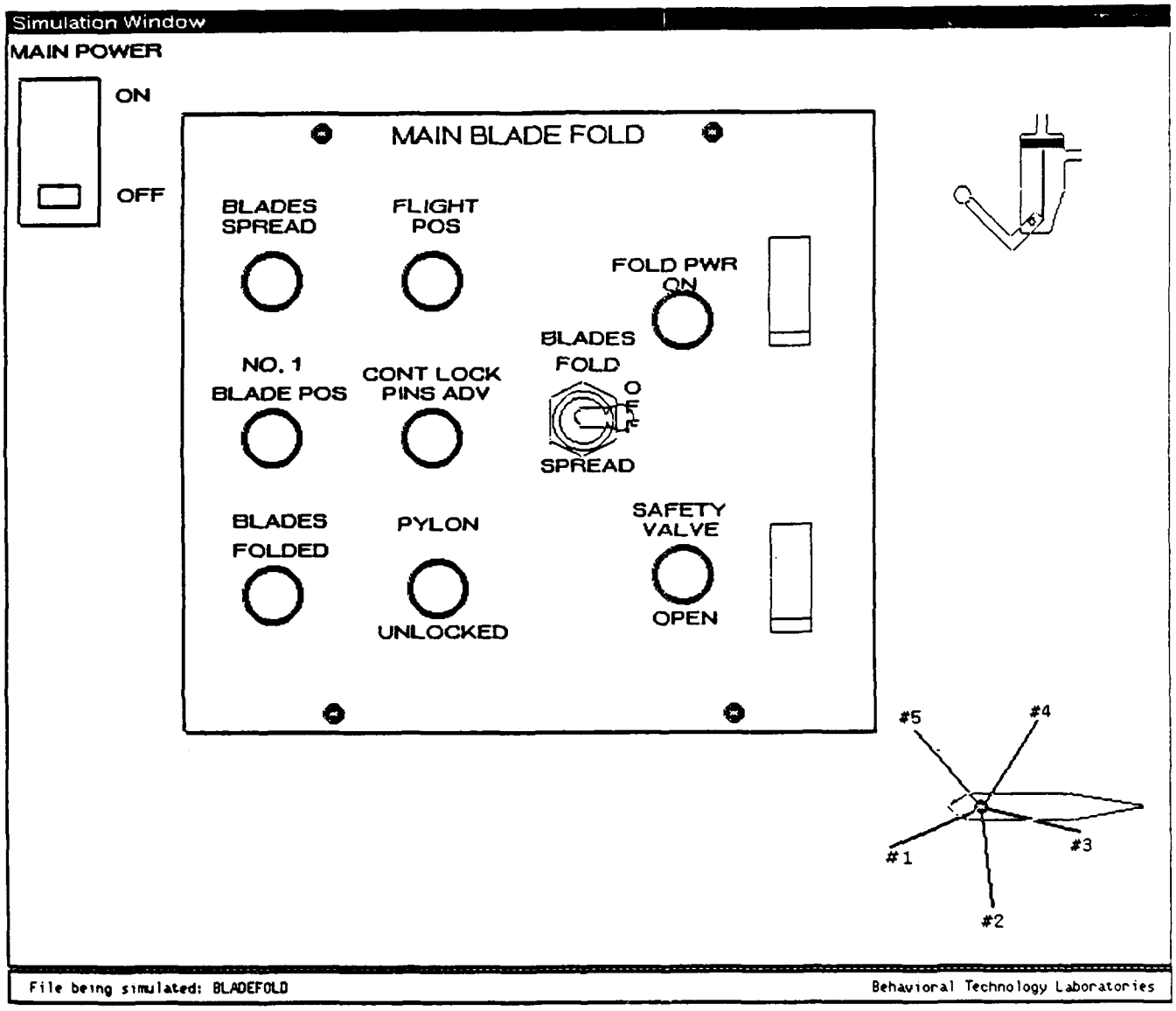
Williams, M. D., Hollan, J. D., and Stevens, A. L. An overview of STEAMER: an advanced computer-assisted instruction system for propulsion engineering. *Behavior Research Methods and Instrumentation*, 1981, **13**, 85-90.

Appendix: Simulations Developed Using IMTS

Two quite large simulations have been developed in IMTS, as described in the text of the report. In addition, a number of simpler simulations have been developed by attendees at IMTS training seminars. Many of these simulations were developed in one or two days, following one or two days of lectures and demonstrations.

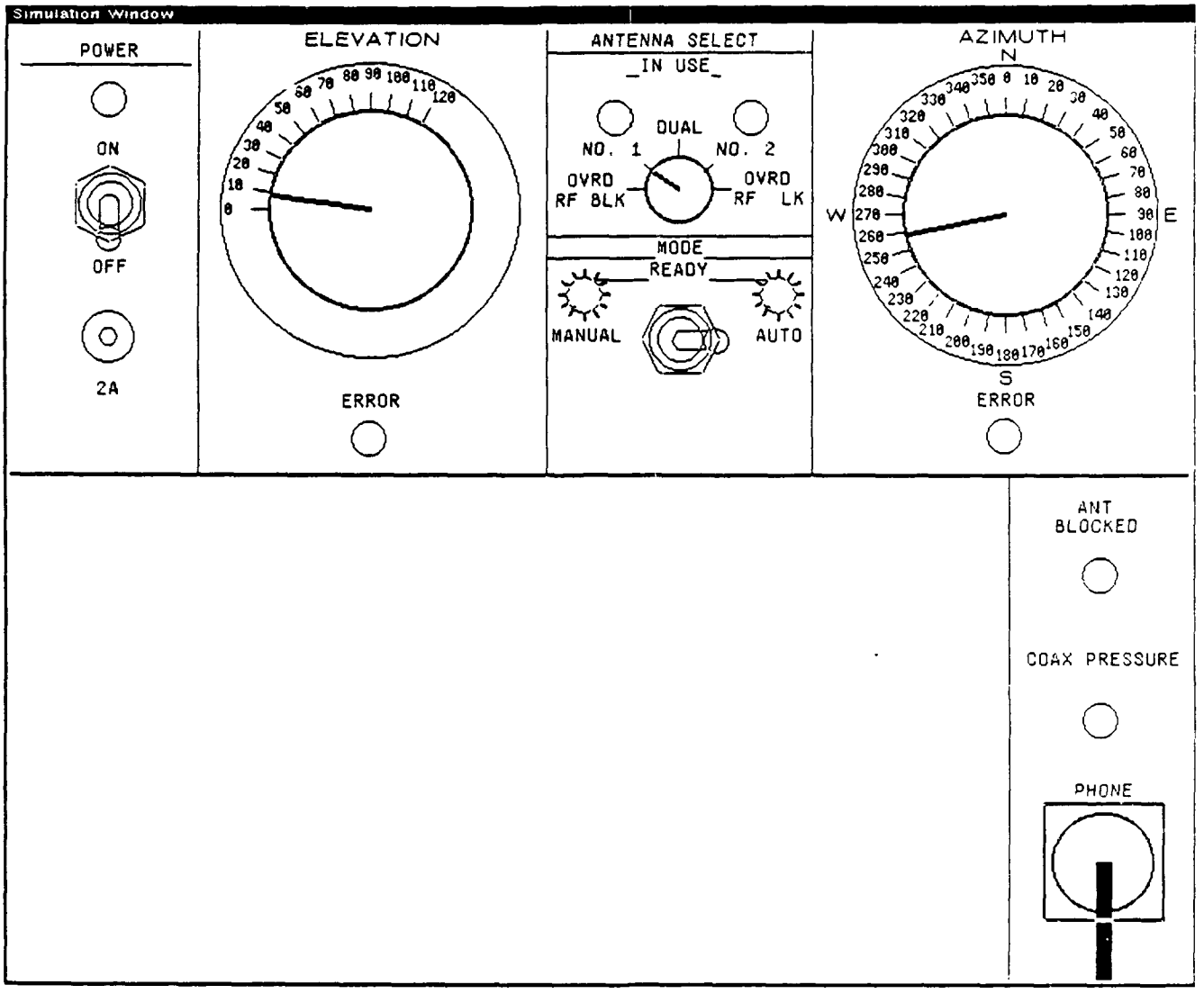


Simulation: SH-3H Helicopter blade folding system
Number of Scenes: 13
Authors: Quentin Pizzini and David Surmon (University of Southern California) with Bill Johnson (Search Technology)

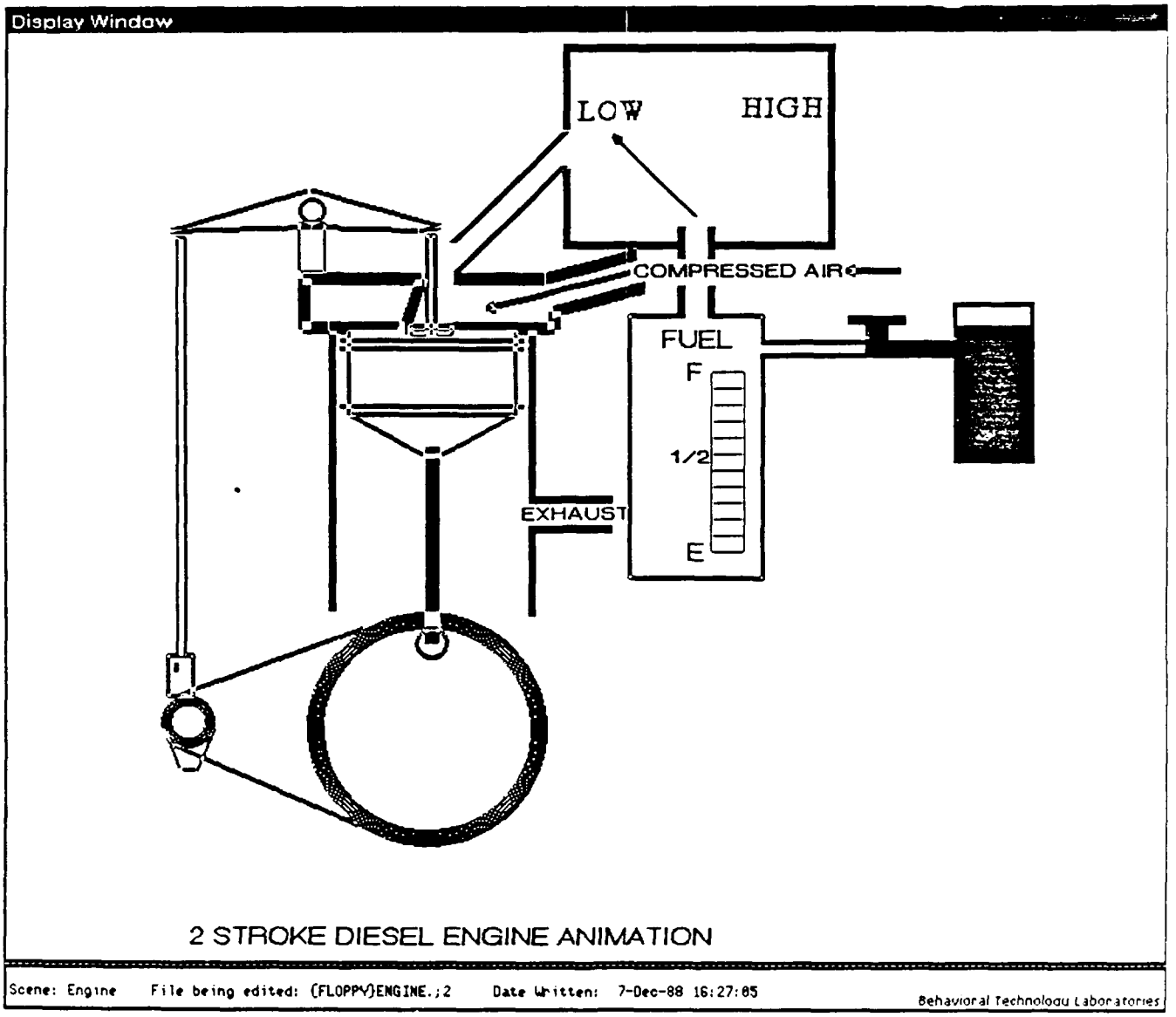


Simulation: SH-3H Helicopter blade folding system — 14th scene
Number of Scenes: 1
Authors: *Vern Malec and Mike Cowan (Navy Personnel Research and Development Center).*

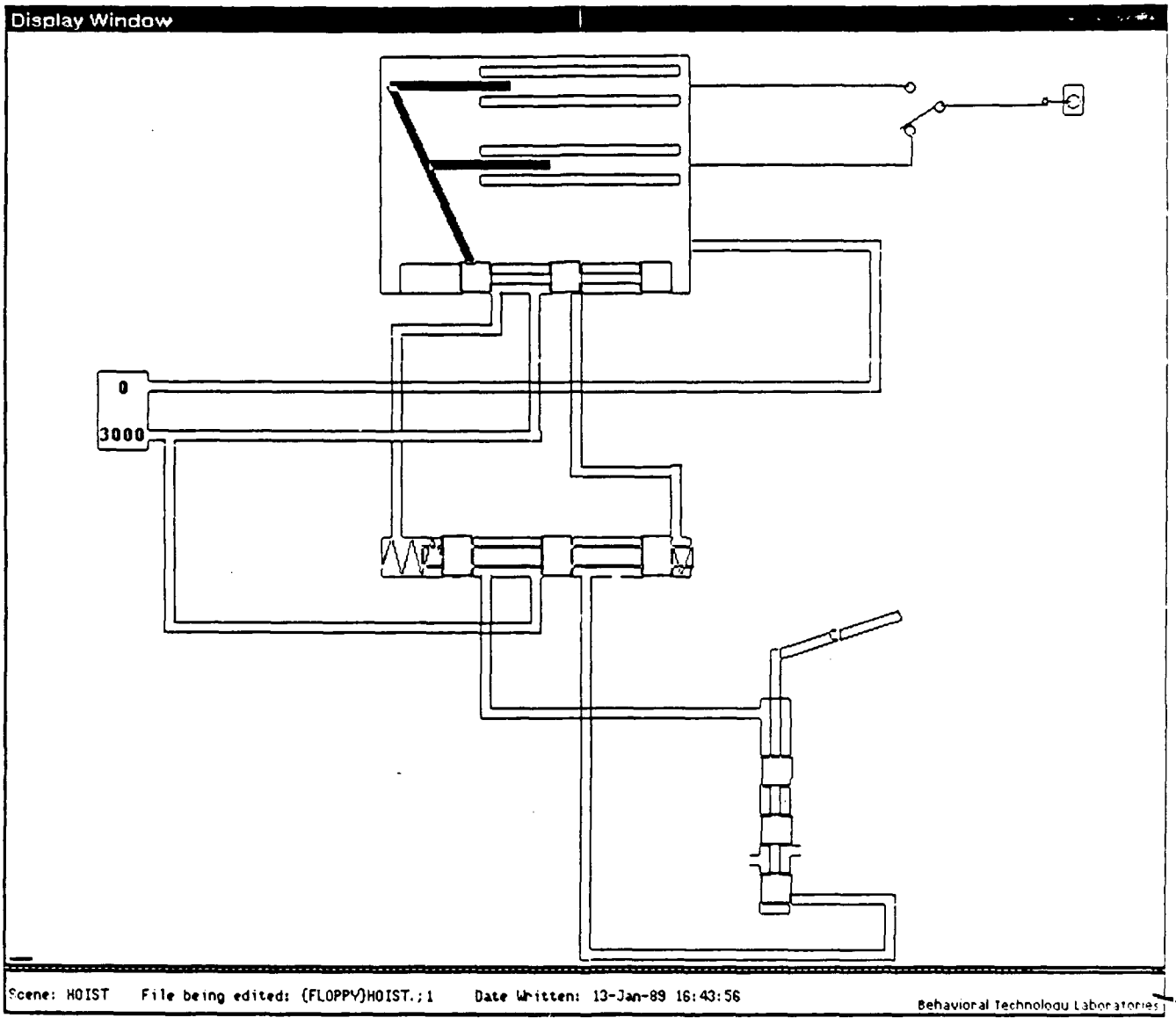
(This fourteenth scene was added to the original thirteen to provide an improved 'front panel' interface. The scene was produced by yoking its objects to objects in the original deep simulation.)



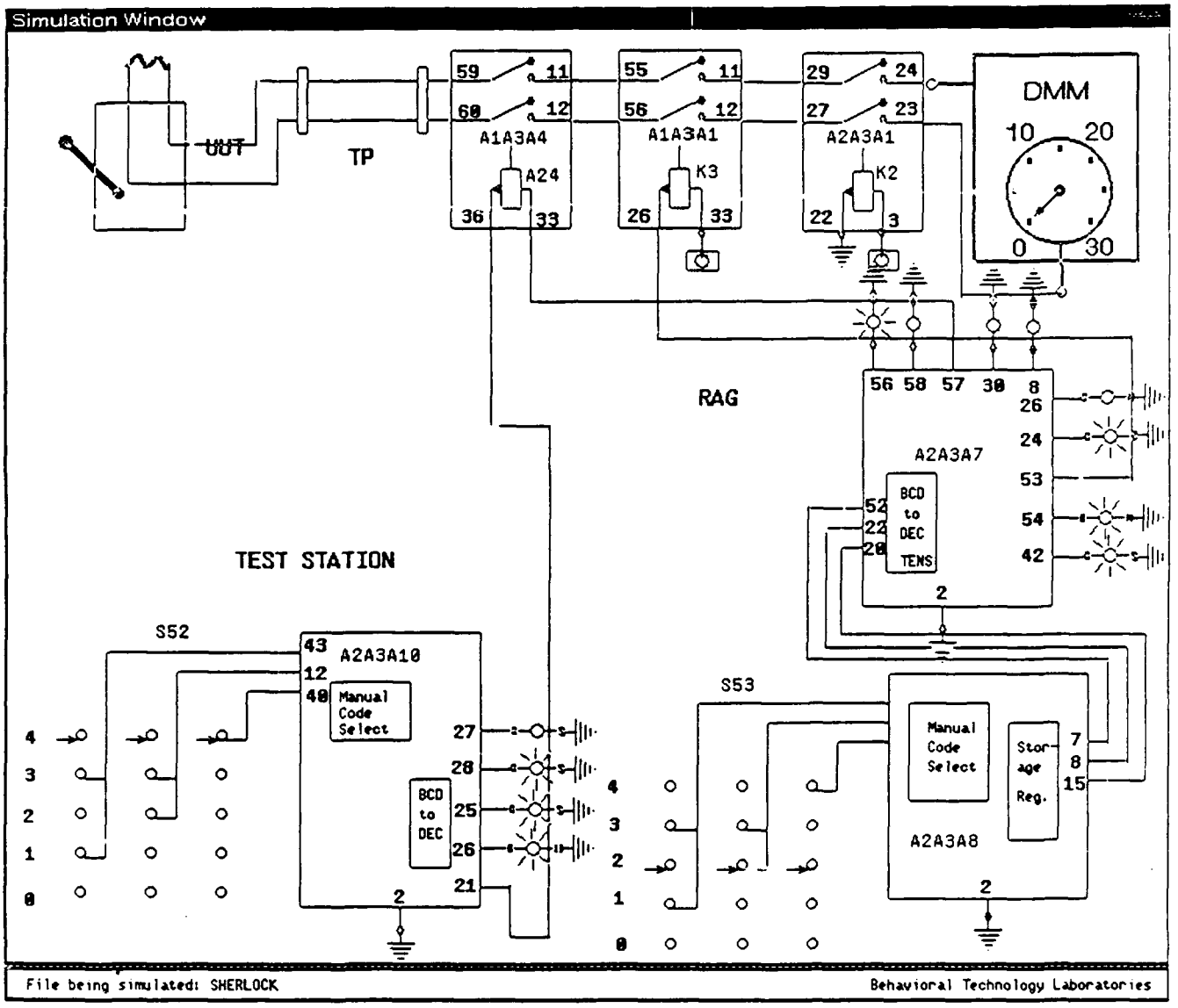
Simulation: WSC-3 Satellite communication system
Number of Scenes: 25
Author: Lee Coller (University of Southern California); data provided by Ron Renfro (Mantech Mathetics).



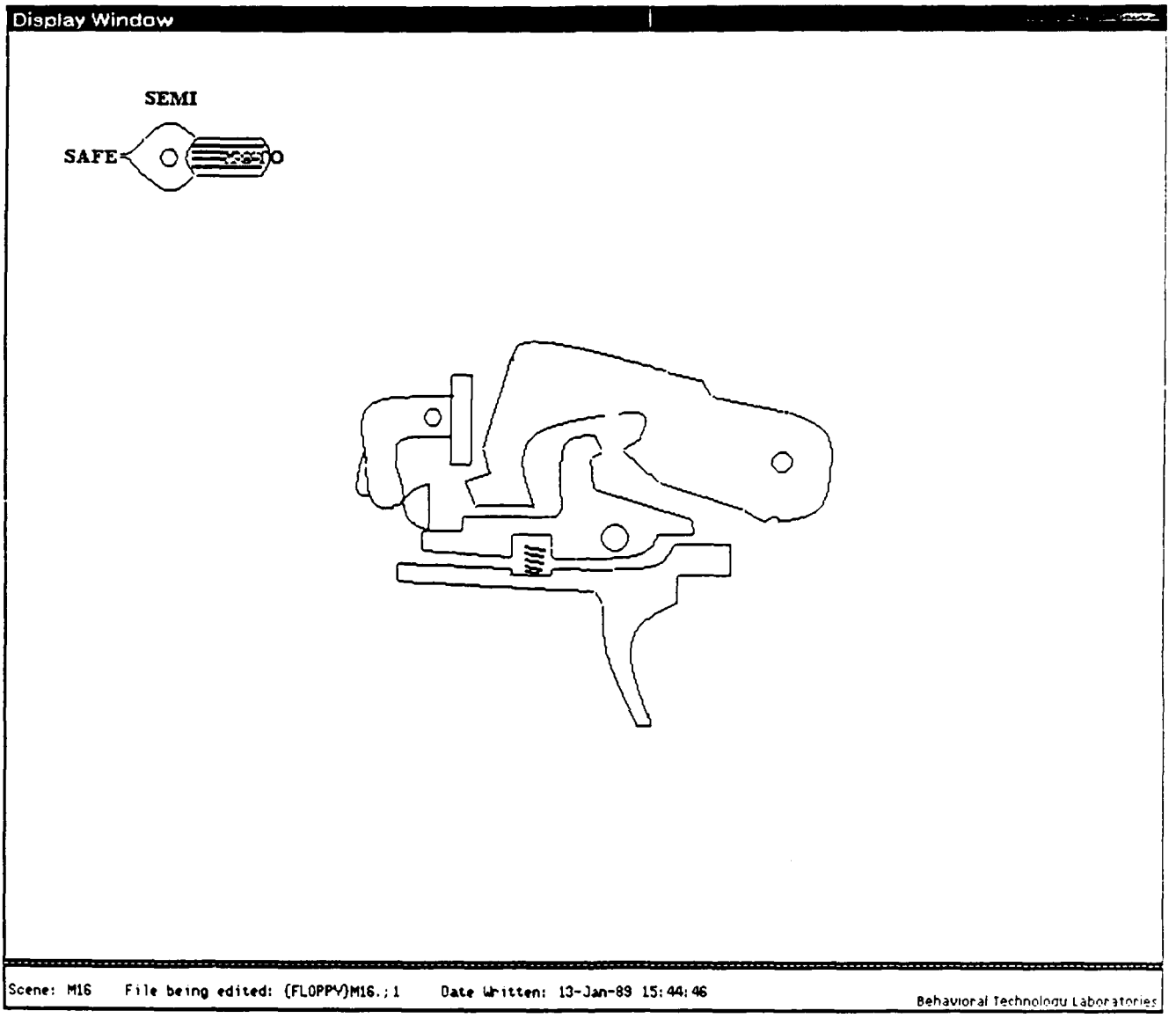
Simulation: Internal combustion engine
Number of Scenes: 1
Authors: Russ Hunt and William Johnson (Search Technology)



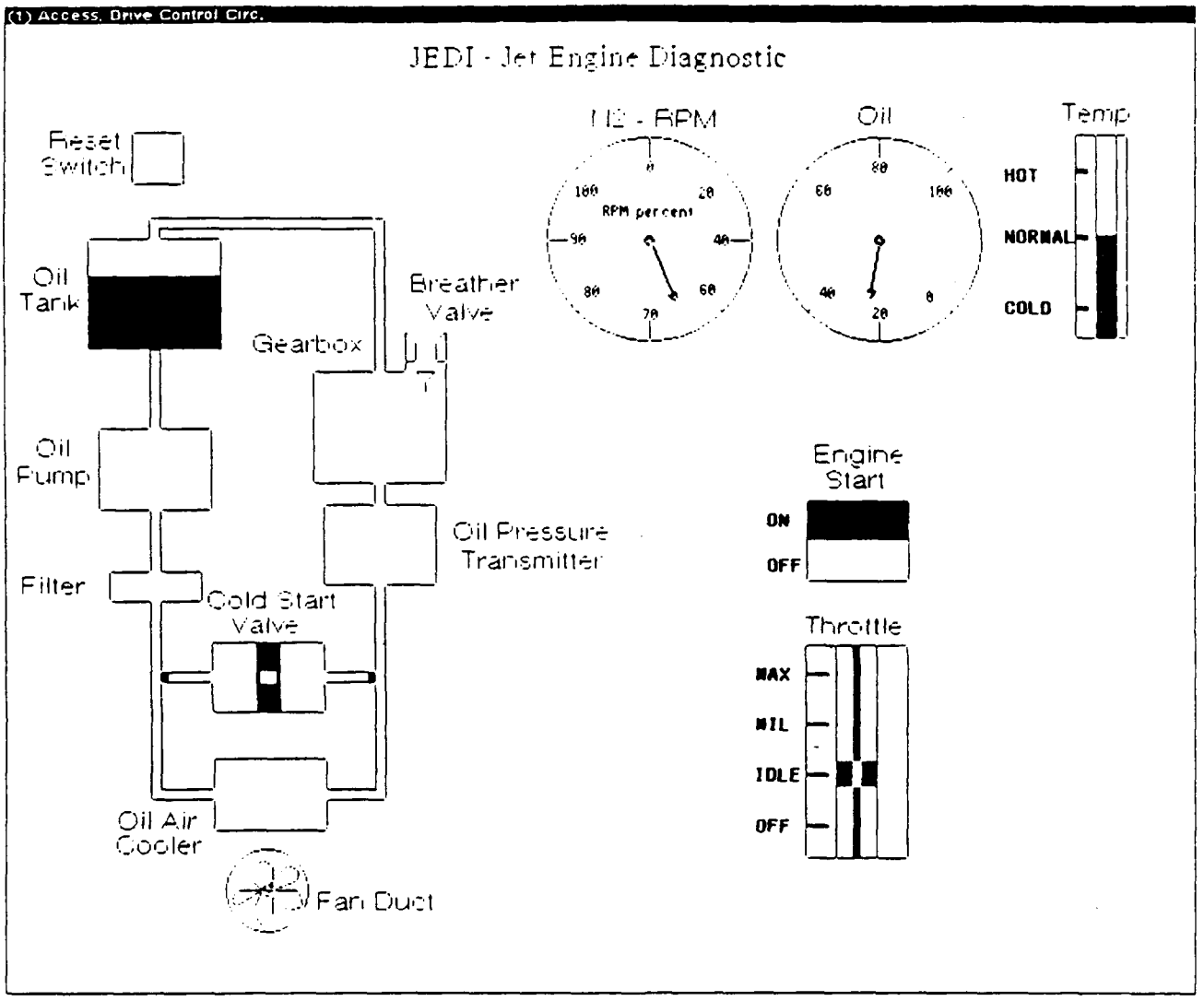
Simulation: Munition Hoist
Number of Scenes: 1
Author: William Murray (FMC)



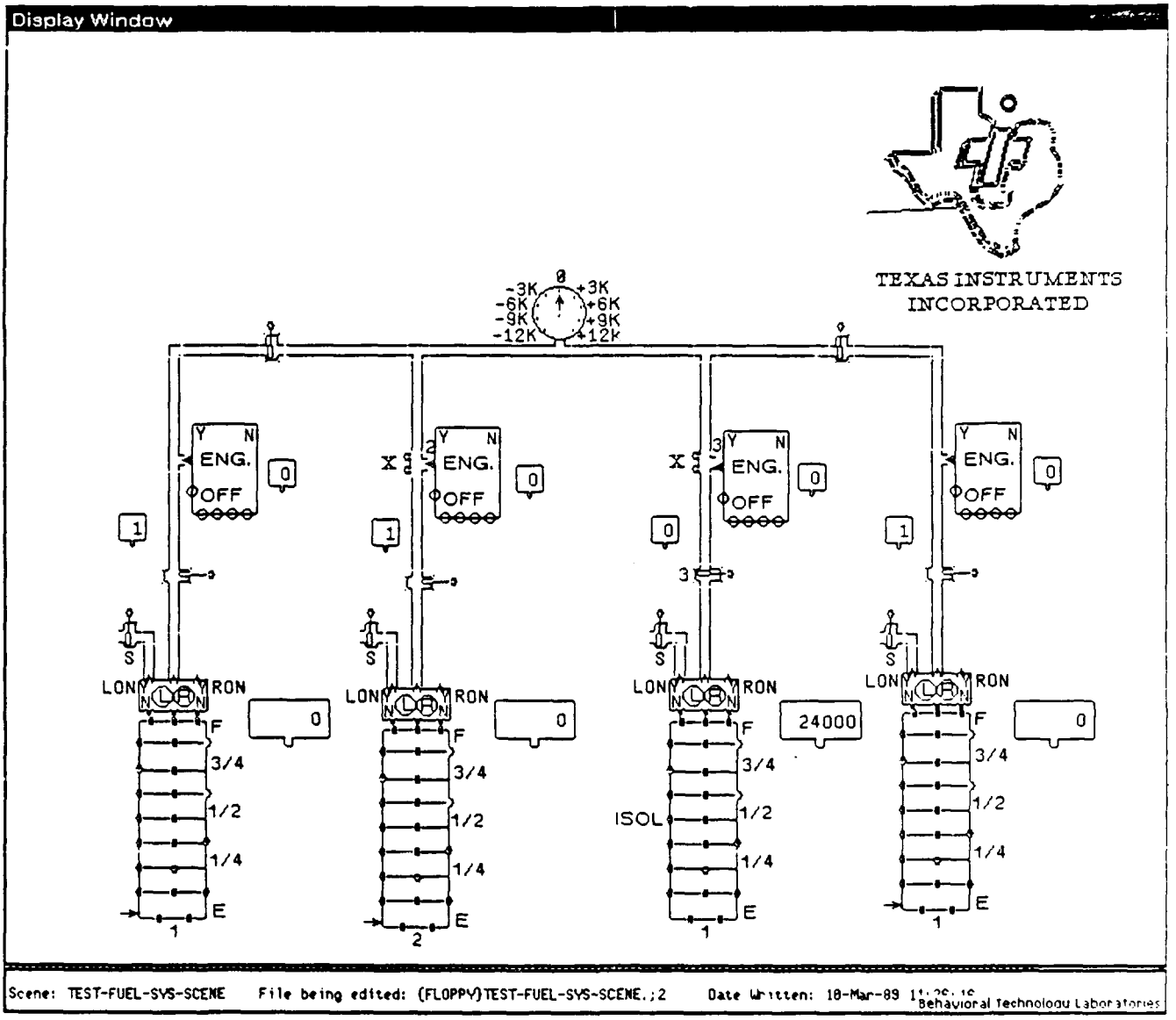
Simulation: F-15 Manual Avionics Test Station
Number of Scenes: 1
Author: Marilyn Bunzo (Learning Research and Development Center, University of Pittsburgh)



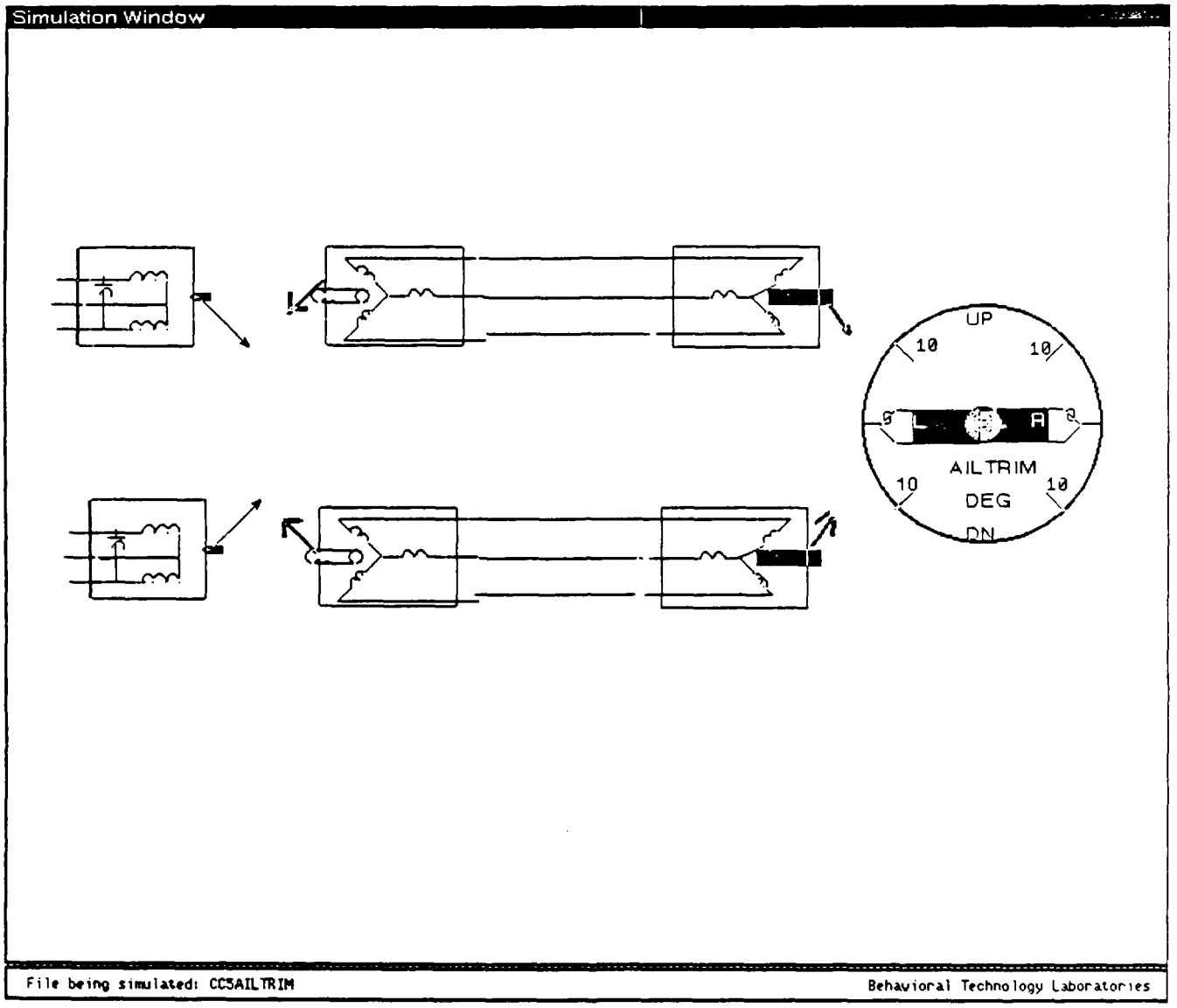
Simulation: M16 Trigger Assembly
Number of Scenes: 1
Author: Randy Morlen (Air Force Human Resources Laboratory)



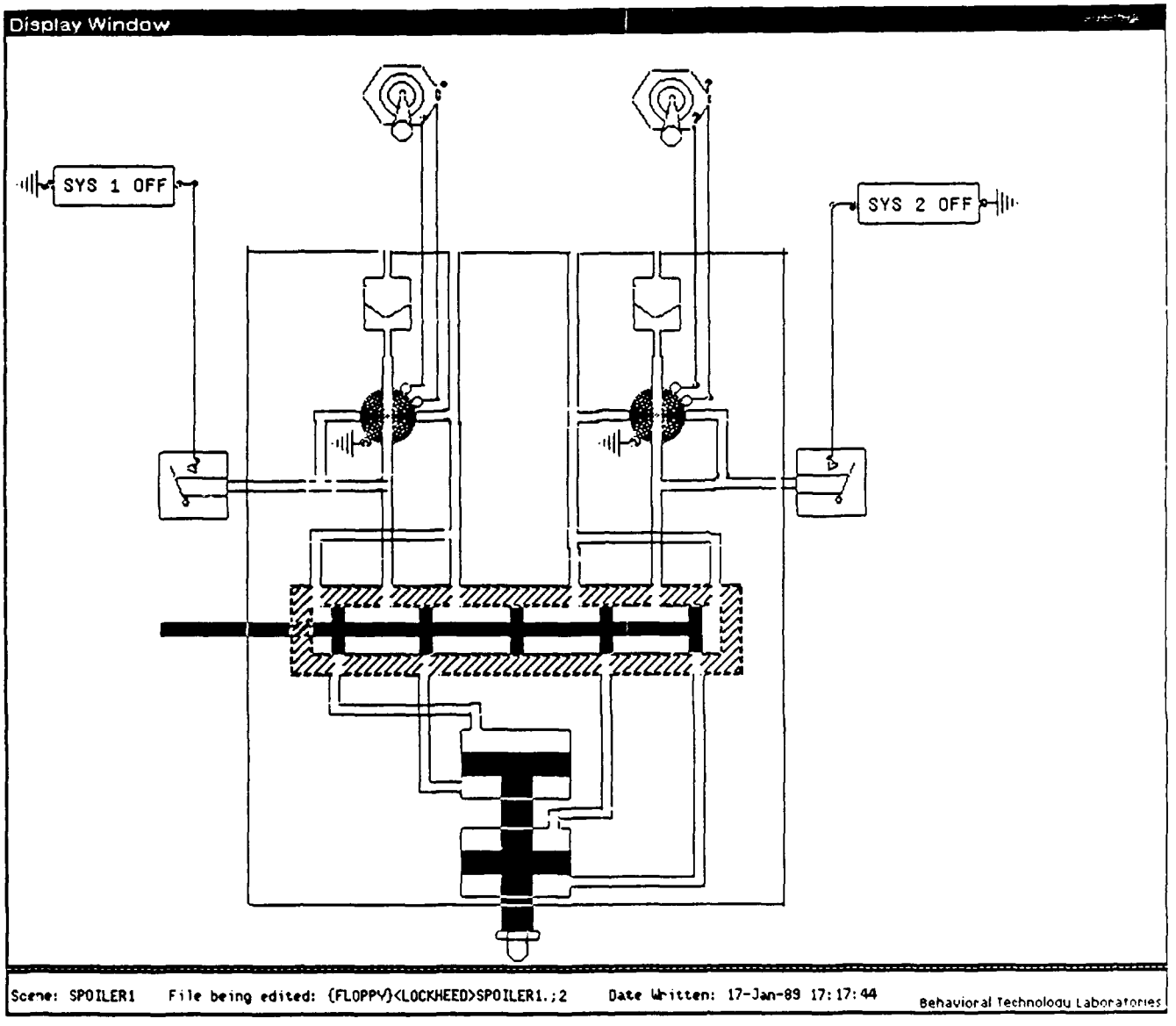
Simulation: Jet engine oil cooling system
Number of Scenes: 1
Author: Quentin Pizzini, David Surmon, Douglas M. Towne, and Allen Munro.
Adapted from:
Keskey, L. C. & Sykes, D. J. Expert systems in aircraft maintenance training. In *Proceedings of the IEEE Weste*
Conference, Anaheim, CA: 1987.



Simulation: Fuel System
Number of Scenes: 1
Authors: Mike Gick (Texas Instruments)



Simulation: C5 Aileron Trim System
Number of Scenes: 2
Authors: Bob Elm and Tom Holzman (Lockheed Georgia)



Simulation: Spoiler
Number of Scenes: 1
Authors: Tom Holzman and Bob Elm (Lockheed Georgia)

Behavioral Technology Laboratory/Towne

Dr. Robert Ahlers
Code N711
Human Factors Laboratory
Naval Training Systems Center
Orlando, FL 32813

Dr. Robert M. Aiken
Computer Science Department
038-24
Temple University
Philadelphia, PA 19122

Dr. Patricia Baggett
School of Education
610 E. University, Rm 1302D
University of Michigan
Ann Arbor, MI 48109-1259

Dr. James D. Baker
Director of Automation and Research
Allen Corporation of America
209 Madison Street
Alexandria, VA 22314

Dr. Meryl S. Baker
Navy Personnel R&D Center
San Diego, CA 92152-6800

Dr. Arthur S. Blaiwes
Code N712
Naval Training Systems Center
Orlando, FL 32813-7100

Dr. Jeff Bonar
Learning R&D Center
University of Pittsburgh
Pittsburgh, PA 15260

LT COL Hugh Burns
AFHRL/IDI
Brooks AFB, TX 78235

Dr. Joanne Capper, Director
Center for Research into Practice
1718 Connecticut Ave., N.W.
Washington, DC 20009

Dr. Ruth W. Chabay
CDEC, Hamburg Hall
Carnegie Mellon University
Pittsburgh, PA 15213

Dr. Fred Chang
Pacific Bell
2600 Camino Ramon
Room 3S-450
San Ramon, CA 94583

Dr. Stanley Collyer
Office of Naval Technology
Code 222
800 N. Quincy Street
Arlington, VA 22217-5000

Dr. Jere Confrey
Cornell University
Dept. of Education
Room 490 Roberts
Ithaca, NY 14853

Dr. Lynn A. Cooper
Department of Psychology
University of Arizona
Tucson, AZ 85747

Dr. Kenneth B. Cross
Anacapa Sciences, Inc.
P.O. Drawer Q
Santa Barbara, CA 93102

Dr. Cary Czichon
Intelligent Instructional Systems
Texas Instruments AI Lab
P.O. Box 660246
Dallas, TX 75266

Brian Dallman
Training Technology Branch
3400 TCHTW/TTGXC
Lowry AFB, CO 80230-5000

Margaret Day, Librarian
Applied Science Associates
P.O. Box 1072
Butler, PA 16003

Dr. Sharon Derry
Florida State University
Department of Psychology
Tallahassee, FL 32306

Behavioral Technology Laboratory/Towne

Defense Technical
Information Center
Cameron Station, Bldg 5
Alexandria, VA 22314
Attn: TC
(12 Copies)

Dr. Thomas M. Duffy
Communications Design
Center, 160 BH
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213

Dr. Pierre Duguet
Organization for Economic
Cooperation and Development
2, rue Andre-Pascal
75016 PARIS
FRANCE

ERIC Facility-Acquisitions
4350 East-West Hwy., Suite 1100
Bethesda, MD 20814-4475

Dr. Debra Evans
Applied Science Associates, Inc.
P. O. Box 1072
Butler, PA 16003

Dr. Beatrice J. Farr
Army Research Institute
PERI-IC
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Elizabeth Fennema
Curriculum and Instruction
University of Wisconsin
225 North Mills Street
Madison, WI 53706

Prof. Donald Fitzgerald
University of New England
Department of Psychology
Armidale, New South Wales 2351
AUSTRALIA

Dr. Michael Flaningam
Code 52
NPRDC
San Diego, CA 92152-6800

Dr. J. D. Fletcher
Institute for Defense Analyses
1801 N. Beauregard St.
Alexandria, VA 22311

Dr. Barbara A. Fox
University of Colorado
Department of Linguistics
Boulder, CO 80309

Department of Humanities and
Social Sciences
Harvey Mudd College
Claremont, CA 91711

Dr. Alinda Friedman
Department of Psychology
University of Alberta
Edmonton, Alberta
CANADA T6G 2E9

Dr. Philip Gillis
Army Research Institute
PERI-II
5001 Eisenhower Avenue
Alexandria, VA 22333-5600

Mr. Lee Gladwin
305 Davis Avenue
Leesburg, VA 22075

Mr. Harold Goldstein
University of DC
Department Civil Engineering
Bldg. 42, Room 112
4200 Connecticut Avenue, N.W.
Washington, DC 20008

Dr. Sherrie Gott
AFHRL/MOMJ
Brooks AFB, TX 78235-5601

Dr. T. Govindaraj
Georgia Institute of
Technology
School of Industrial
and Systems Engineering
Atlanta, GA 30332-0205

Behavioral Technology Laboratory/Towne

Dr. Dik Gregory
Admiralty Research
Establishment/AXB
Queens Road
Teddington
Middlesex, ENGLAND TW110LN

Michael Habon
DORNIER GMBH
P.O. Box 1420
D-7990 Friedrichshafen 1
WEST GERMANY

Dr. Henry M. Halff
Halff Resources, Inc.
4918 33rd Road, North
Arlington, VA 22207

Mr. H. Hamburger
Department of Computer Science
George Mason University
Fairfax, VA 22030

Dr. Cheryl Hamel
NTSC, Code 711
Orlando, FL 32813

Janice Hart
Office of the Chief
of Naval Operations
OP-111J2
Department of the Navy
Washington, D.C. 20350-2000

Dr. Wayne Harvey
Center for Learning Technology
Education Development Center
55 Chapel Street
Newton, MA 02160

Dr. James E. Hoffman
Department of Psychology
University of Delaware
Newark, DE 19711

Ms. Julia S. Hough
110 W. Harvey Street
Philadelphia, PA 19144

Dr. Steven Hunka
3-104 Educ. N.
University of Alberta
Edmonton, Alberta
CANADA T6G 2G5

Mr. Roland Jones
Mitre Corp., K-203
Burlington Road
Bedford, MA 01730

Dr. Marcel Just
Carnegie-Mellon University
Department of Psychology
Schenley Park
Pittsburgh, PA 15213

Dr. Michael Kaplan
Office of Basic Research
U.S. Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333-5600

Dr. David Kieras
Technical Communication Program
TIDAL Bldg., 2360 Bonisteel Blvd.
University of Michigan
Ann Arbor, MI 48109-2108

Dr. Lois-Ann Kuntz
3010 S.W. 23rd Terrace
Apt. No. 105
Gainesville, FL 32608

Dr. Doris K. Lidtke
Software Productivity Consortium
1880 Campus Commons Drive, North
Reston, VA 22091

Dr. Robert Lloyd
Dept. of Geography
University of South Carolina
Columbia, SC 29208

Dr. Jack Lochhead
University of
Massachusetts
Physics Department
Amherst, MA 01003

Vern M. Malec
NPRDC, Code 52
San Diego, CA 92152-6800

Behavioral Technology Laboratory/Towne

Dr. James McMichael
Technical Director
Navy Personnel R&D Center
San Diego, CA 92152-6800

Dr. Arthur Melmed
Computer Arts and
Education Laboratory
New York University
719 Broadway, 12th floor
New York, NY 10003

Dr. Vittorio Midoro
CNR-Istituto Tecnologie Didattiche
Via All'Opera Pia 11
GENOVA-ITALIA 16145

Dr. Jason Millman
Department of Education
Roberts Hall
Cornell University
Ithaca, NY 14853

Dr. Lynn Misselt
HQM-222
Control Data Corporation
Box 0
Minneapolis, MN 55440

Dr. Andrew R. Molnar
Applic. of Advanced Technology
Science and Engr. Education
National Science Foundation
Washington, DC 20550

Dr. William Montague
NPRDC Code 13
San Diego, CA 92152-6800

Dr. William R. Murray
FMC Corporation
Central Engineering Labs
1205 Coleman Avenue
Box 580
Santa Clara, CA 95052

Dr. Harold F. O'Neil, Jr.
School of Education - WPH 801
Department of Educational
Psychology & Technology
University of Southern California
Los Angeles, CA 90089-0031

Office of Naval Research,
Code 1142CS
800 N. Quincy Street
Arlington, VA 22217-5000
(6 Copies)

Dr. Judith Orasanu
Basic Research Office
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Nancy N. Perry
Naval Education and Training
Program Support Activity
Code-047
Building 2435
Pensacola, FL 32509-5000

Dept. of Administrative Sciences
Code 54
Naval Postgraduate School
Monterey, CA 93943-5026

Dr. Joseph Psotka
ATTN: PERI-IC
Army Research Institute
5001 Eisenhower Ave.
Alexandria, VA 22333-5600

Dr. J. Wesley Regian
AFHRL/IDI
Brooks AFB, TX 78235

Dr. Charles M. Reigeluth
330 Huntington Hall
Syracuse University
Syracuse, NY 13244

Dr. Daniel Reisberg
Reed College
Department of Psychology
Portland, OR 97202

Mr. William A. Rizzo
Code 71
Naval Training Systems Center
Orlando, FL 32813

Behavioral Technology Laboratory/Towne

Dr. Linda G. Roberts
Science, Education, and
Transportation Program
Office of Technology Assessment
Congress of the United States
Washington, DC 20510

Dr. Janet W. Schofield
816 LRDC Building
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15260

Dr. Judith W. Segal
OERI
555 New Jersey Ave., NW
Washington, DC 20208

Dr. Robert J. Seidel
US Army Research Institute
5001 Eisenhower Ave.
Alexandria, VA 22333

Dr. Randall Shumaker
Naval Research Laboratory
Code 5510
4555 Overlook Avenue, S.W.
Washington, DC 20375-5000

Dr. Derek Sleeman
Computing Science Department
The University
Aberdeen AB9 2FX
Scotland
UNITED KINGDOM

Ms. Gail K. Slemon
LOGICON, Inc.
P.O. Box 85158
San Diego, CA 92138-5158

Dr. Alfred F. Smode
Code 7A
Research and Development Dept.
Naval Training Systems Center
Orlando, FL 32813-7100

Dr. Elliot Soloway
Yale University
Computer Science Department
P.O. Box 2158
New Haven, CT 06520

Linda B. Sorisio
IBM-Los Angeles Scientific Center
11601 Wilshire Blvd., 4th Floor
Los Angeles, CA 90025

Dr. Marian Stearns
SRI International
333 Ravenswood Ave.
Room B-5124
Menlo Park, CA 94025

Dr. Saul Sternberg
University of Pennsylvania
Department of Psychology
3815 Walnut Street
Philadelphia, PA 19104-6196

Dr. David E. Stone
Computer Teaching Corporation
1713 South Neil Street
Urbana, IL 61820

Dr. Perry W. Thorndyke
FMC Corporation
Central Engineering Labs
1205 Coleman Avenue, Box 580
Santa Clara, CA 95052

Dr. Douglas Towne
Behavioral Technology Labs
University of Southern California
1845 S. Elena Ave.
Redondo Beach, CA 90277

Dr. Zita E. Tyer
Department of Psychology
George Mason University
4400 University Drive
Fairfax, VA 22030

Dr. Frank L. Vicino
Navy Personnel R&D Center
San Diego, CA 92152-6800

Dr. Jerry Vogt
Navy Personnel R&D Center
Code 51
San Diego, CA 92152-6800

Dr. Thomas A. Warm
Coast Guard Institute
P. O. Substation 18
Oklahoma City, OK 73169

Behavioral Technology Laboratory/Towne

Dr. Beth Warren
BBN Laboratories, Inc.
10 Moulton Street
Cambridge, MA 02238

Dr. Shih-sung Wen
Department of Psychology
Jackson State University
1400 J. R. Lynch Street
Jackson, MS 39217

Dr. Douglas Wetzel
Code 51
Navy Personnel R&D Center
San Diego, CA 92152-6800

Dr. Barbara White
BBN Laboratories
10 Moulton Street
Cambridge, MA 02238

Dr. Marsha R. Williams
Applic. of Advanced Technologies
National Science Foundation
SEE/MDRISE
1800 G Street, N.W., Room 635-A
Washington, DC 20550

Dr. Robert A. Wisher
U.S. Army Institute for the
Behavioral and Social Sciences
5001 Eisenhower Avenue
Alexandria, VA 22333-5600

Dr. Merlin C. Wittrock
Graduate School of Education
UCLA
Los Angeles, CA 90024

Dr. Wallace Wulfeck, III
Navy Personnel R&D Center
Code 51
San Diego, CA 92152-6800

Dr. Masoud Yazdani
Dept. of Computer Science
University of Exeter
Prince of Wales Road
Exeter EX44PT
ENGLAND

Dr. Joseph L. Young
National Science Foundation
Room 320
1800 G Street, N.W.
Washington, DC 20550